School-Based Assessment 2021

Information and Communication Technology

Hartanto Kwee, Jeffrey 5A (20)

15th July, 2020

0. TABLE OF CONTENTS

1. DESIGN

A. Game Definition

The name of the game is THE BECOMING OF A MATHEMATICIAN. The game is an **educational game** in **Mathematics** taking the form of an **RPG (Role-Playing Game)**[1].

B. Target Users

The target users are mainly senior secondary school students. Ambitious students from lower grades may also challenge themselves using this game.

C. Area of Learning

Mathematics is a difficult subject to learn, mainly because it takes energy and time to understand and remember various formulae. It is also partly because of the cumulative characteristic of studying Mathematics – advanced topics are built on top of simpler concepts. One would have to master a preceding concept to smoothly move on to harder ones. Therefore, *practice is very important in Mathematics*, as it helps the mind to quickly recite an equation which is crucial in solving higher-order problems.

This game helps students spice up the fun in the tiring process of practicing Mathematics by rewarding them with virtual money and experience that indicates their progress in this field of study.

People are more motivated and encouraged to do something if they are able to objectively view their progress. As players fight more monsters and Mathematicians in the game by completing Math problems, and gain more in-game currency and experience, they can grow stronger by buying equipment and developing skills. This strength is concrete evidence that players are able to tackle problems, and the feeling of power is the perfect reward to them.

This game will provide students with a diverse database of questions in order to help them learn and practice Mathematics. The area of learning is divided into three levels of difficulty, namely[2]:

Level 1: Quadratic Equations and Graphs

Level 2: Arithmetic Sequences

Level 3: Geometric Sequences and Polynomials

[1] An RPG is a game in which players assume the roles of characters in a fictional setting. (Adapted from Wikipedia, "Role-playing game". Accessed 15th March, 2020.)

[2] Originally, inequalities (linear and quadratic), equations of straight Line and circles and permutation and combination were added. Due to time limitations, they were removed. However, if a system that can read any type of question is developed as long as the question can be solved by substituting coefficients, there can be however many questions. An interface for students to enter new equations may also be developed based on this convenient system.

D. Game Rules

There are three modes in the game, namely the Adventure Mode, the Multiplayer Mode and the Practice Mode.

1. *Adventure Mode*

The Adventure Mode is a role-playing game, where the player becomes the hero (or heroine) of a virtual world to experience the story.

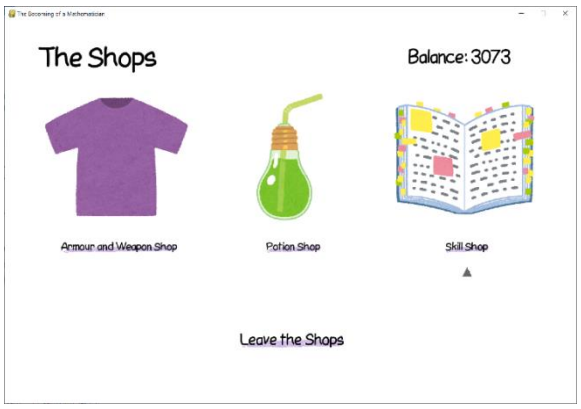
Storyboard: The story takes place in a small village. One day, your parents brought you out to town to look around. You entered a bookshop and found an interesting book filled with eccentric but beautiful symbols. The title of the books was called "Mathematics". Your desire to learn resonated, and you started learning and learning, discovering the impressive work of famous Mathematicians. As you dug deeper down into the grounds of Mathematics, your admiration grew along with the passion to meet these Mathematicians yourselves. *You then set your eyes on a final goal: to defeat these Mathematicians and to claim the position of being the best Mathematician*. Just like so, you embarked on a journey to travel around the world to defeat these Mathematicians.

Gameplay: *The goal in Adventure Mode is to defeat every Mathematician in the game. There are a total of three "levels", each with a different category of Math questions, and in each level there are two Mathematicians.*

The game officially starts at the **Village**. At the Village's selection menu, you can access all features in Adventure Mode. First of all, the "Math Guide" gives the player an overview on the types of questions covered in the level, and provides web page links to useful website for revision and further learning.

▲ **Fighting a Mathematician.**

Throughout the game, you will have to fight opponents to strengthen yourself. You can fight civilians to level up by choosing **"Grinding"**, or to fight Mathematicians, select **"Journey On..."**. During a fight, you have answer Math questions in order to deal damage to your opponent, and after your attack your opponent will also attack back. After you have completed a fight, you will gain money and experience points (EXP).



▲ **The Shops.**

The player has an attribute called "level". The more he fights, the more EXP he will gain, and when the amount of EXP reaches a certain amount, the player will "level up" and gain something called Skill Points.

Money and Skill Points can be spent in **"The Shops"**. There are three shops in the game: "Armor and Weapon Shop" (selling armor and weapons that permanently increase defense and attack), "Potion Shop" (selling potions that

temporary boosts performance) and “Skill Shop” (“skills” that can permanently improve performance).

I*nformation about the effects of skills and equipment can be found in “View Statistics”*, which can be accessed from the **“Game Menu”**. From there, it is also options to save and quit the game.

2. Multiplayer Mode

In **Multiplayer Mode**, two players will battle each other. There are two modes: Offline and Online Mode.

In **Offline Multiplayer mode**, two players use the same computer and take turn to attack each other.

In **Online Multiplayer mode**, the two players will connect through the internet and fight each other. The players will first choose a category of question to answer. Each category will deal different magnitudes of damage. The players may answer the question at any time, and once the question is answered correctly, an attack will be immediately made. After the attack, the player may immediately start another round of attack without having to wait for his opponent to finish. The player who defeats the other the quickest wins.

3. Practice Mode[3]

In the Practice Mode, players will be able to directly access the question generator. There is no time limit, and players can choose what category of questions they would like to do.

E. Input and Output

There are several input and output data in the game (input and output files will be discussed in global variables), and their corresponding devices are required in order to perceive them. These devices include **a keyboard** (essential), **a mouse** (optional), **a Visual Display Unit** (i.e. monitor) and **speakers**. The I/O sources are summarized below:

1. Input

Text: When answering a question, the player is required to type in their answers. In the shop, the player needs to specify the quantity of a product he would like to purchase. When playing online multiplayer mode, the player needs to type the IP address in order to join a server. Therefore, a keyboard is imperative to gameplay.

Selection Menus: Selection menus can be traversed just by using the keyboards’s arrow buttons, so the keyboard is the most important input device. In addition, the mouse (a pointing device) can aslo be used to click on choices, and is considered a secondary input device[4].

[3] It is also called Debug Mode.
[4] The game was originally menu driven, but has been changed into a GUI.

2. Output

Graphics: Game elements, like selection menus, battle interfaces and Mathematics questions are all displayed graphically, so a monitor is required to play the game.

Sound: The game has sound effects (button clicking sounds) and background music, so speakers or headphones may be used to play sound.

F. Data Structures

1. Array-Like Data Types

Python has special array-like data types when compared with conventional programming languages (like Pascal and C), which are unexpectedly versatile and useful.

List, Tuples: same as conventional arrays, but their indices start from 0, and negative indices *–n* specifies the *n* element counting from the end of the list. *Use*: They are used to store file paths to images for the Entrance Sequence and the Epilogue, to display them one by one. They are also used to store choices in selection menus.

Dictionary: contains 'key' and 'value' pairs. Values are assigned to a unique key in the dictionary, and the values are accessed by specifying the key. *Use*: Dictionaries are directly translated to key-value pairs in JSON, which is used to store game data (questions and game state) in files.

Set: similar to lists and tuples, except its values do not repeat and its elements have no definite ordering. *Use*: Since it is possible to convert from one array type to another, the expression `list(set(X))` can be used to eliminate repeated elements from the list `X`.

```
L = [-1, -2, -3, -4] # A List
T = (1, 3, 5, 7) # A Tuple
for i in L + list(T):
    print((tuple(L) + T)[i], end=" ")
# Output: 7 5 3 1 -2 -4 3 7
```

▲ **Lists and Tuples**

```
# List-Set Conversion
L = [1, 3, 5, 2, 4, 6, 1, 3]
S = set(L)
print(S) # Output: {1, 2, 3, 4, 5, 6}
L = list(S)
print(L) # Output: [1, 2, 3, 4, 5, 6]
```

▲ **Sets**

```
# Dictionaries
T = {'a': 3, 3+2j: 4.13, 'string': [1,3]}
print(T['a']) # 3
print(T[3+2j]) # 4.13
print(T['string']) # [1, 3]
```

▲ **Dictionaries**

2. *Object Oriented Programming and Dictionaries*

Python also supports object orientated programming, so another data structure that can be used is objects. Objects contain attributes and methods, making objects a more structured and compact piece of abstraction that encapsulates and hides its implementation.

```
class Coconut:
    def __init__(self):
        self.one = 1
        self.two = 2
        self.three = 3
```

▲ **Defining a Class**

Since dictionaries have similar structure in that key-value pairs are similar to attributes in objects, it was sometimes necessary to decide which one to use. To select a suitable data type, we should first look at the characteristics and benefits of adopting each data structure.

(Dictionaries and Objects are equally as good:)

i. **Memory Usage**[5]: In general, normal objects will take more time to process than dictionaries. This makes dictionary a lightweight choice for generating many of copies of the same storage structure. On the other hand, the `__slots__` attribute, when added to the definition of an object, states the number of attributes that the object has such that just enough resources are allocated for its instances. Using this, objects may be considerably better than dictionaries.

(Dictionaries are better:)

ii. **Dictionary Iteration**: Dictionaries can be put into `for` loops to access every one of the key-value pairs in it, which is not possible for classes. It is easier to access its elements.

iii. **Dictionary Keys**: Dictionary keys can be anything, from characters to integers to even complex numbers. Therefore, when a long list of items has to be sorted into several groups, dictionaries should be used.

`nut = Coconut()`
`print(nut.one, nut.two, nut.three)`
`# 1 2 3`
	`CoconutTree = {1: 1, 2: 2, 3: 3}`
	`for i in range(1, 3+1):`
	`  CoconutTree[i] = i`
	`print(CoconutTree) # {1: 1, 2: 2, 3: 3}`

Class	**Dictionary**

▲ **Iteration using a dictionary, and typing class attributes manually.**

[5] Reference: *https://stackoverflow.com/questions/1336791/dictionary-vs-object-which-is-more-efficient-and-why*.

(Objects are better:)

iv. **Error-proof**: Classes are more error-proof than dictionaries, since a wrong attribute name will immediately raise an error at compile time, and the IDE will autocomplete the attribute name for me. However, dictionary keys are only validated during run time, so it will raise a run time error if the key specified does not have a corresponding value. It is sometimes very annoying to debug.

`nut = Coconut()`
`print(nut.zero, nut.two, nut.three)`
`# AttributeError: 'Coconut' object has no attribute 'zero'`
	`for i in range(3):`
	`  print(CoconutTree[i])`
	`# KeyError: 0`

Class	**Dictionary**

▲ Dictionary generates run-time errors, while classes generates syntax errors

v. **Structure**: Classes have more structure than dictionaries. First, classes contain their own methods, so as long as the file containing the declaration of the class is imported, these functions can be accessed through the object. Dictionaries do not have such functionality, so it is seemingly less structured. Second, classes have their own "data types", while dictionary don't. So, all dictionaries would seem like the same as others, even though they store different types of data.

vi. **Self-initiation**: When an object is defined, its attributes can automatically be generated and initiated using parameters given on declaration through its constructor method, instead of having to rely on an external function to generate a dictionary.

Based on the above discussion, the following shows how these two structures are used[6]:

Dictionary	● Question Objects ➢ ✓ JSON ● Question List [i.e. Dictionary of Dictionaries] ➢ ✓ Iteration: sorting according to difficulty, ✓Keys ● Boss List [i.e. Dictionary of Objects] ➢ ✓ Iteration, ✓Keys
Object	● Player Data ➢ ✓✓ Methods, ✓ Initiation, ✓ Errors ● Mathematical Objects ➢ ✓✓ Methods, ✓ Initiation
`__slots__` Object (✓ Size)	● Boss Entries ➢ ✓ Methods, ✓ Errors ● "Sprites" ➢ ✓ Initiation, Methods, ✓ Errors

[6] ✓ denotes the main deciding factors when the data structure is chosen, and shaded items are those which were of largest debate of what data structure to use.

3. Question Generators

In the present implementation of the question generator (refer to the program documentation), a database is used to store questions, and the generator is responsible for generating valid coefficients to create a question. After extracting the question statements from a file, they will be sorted in a dictionary with difficulty as the key.

The debate lies on what data type should the question and answer statements be in. Since there are placeholders in the unprocessed question statement for inserting the coefficients, there are three types of data storage structures that can be chosen from in this occasion: strings (array of characters), array of strings and linked lists (of characters). Let's list out their pros and cons:

	Pros (Advantages)	**Cons (Disadvantages)**
Strings	♦ Random access is possible ♦ Tokenization (pattern-matching) is easier	♦ It would take a bit more time as every single character has to be read and processed every time a question is generated.
Linked Lists of Characters	♦ Shifting is not needed when replacing stand-in substrings with coefficients	♦ Memory-inefficient ♦ Only sequential access is possible. Searching would take time.
Array of Strings	♦ Placeholders are already separated out, saving time to identify placeholders every time.	♦ A less flexible choice to store a string. ♦ Needs an identifier to tell whether the array element is normal text or asking for an attribute.

Linked lists are obviously the worst choice possible, not only is it memory-inefficient, it is also time-consuming. The "shifting" concern is also not justified, as this is just caused by viewpoint of strings as "character arrays". In reality, strings can be cut and rejoined, so shifting isn't really needed.

Array of strings and strings are both very effective[7]. However, at last, strings are chosen because of its simplicity of implementation, and ease of creating questions[8].

[7] In fact, array of strings should be the best choice in terms of program efficiency, but this idea only came to mind after the data structure was decided. This is an improvement to be made.

[8] This is because there is no need to separate the question into array elements. It is also easier to edit the question since the question is more readable than an array.

4. Loading Images and Fonts

Images are used throughout the program, and there are many images that have to be loaded throughout the execution of the game. When an image is loaded (by the `pygame` module), it is stored in the program in a type of object called a `Surface`. The memory usage of all of these Surfaces can be considerably large when many of them are loaded.

At program start-up, **global variables are created for each image to either store the file path (a string) to the image file for loading later, or the Surface object after loading the image at program start up**. This is to reduce effort when editing images used by the program: just changing the content of the global variables will change every use of that image in the program at once.

Fonts are loaded in a similar way, except they are stored in a `Font` object.

The following factors are considered when choosing what data type to use

1. **Frequency of usage**: if the image is frequently used, like arrows and labels in selection menus, then they should be stored as Surfaces, as loading them frequently would generate delay. If it is less frequently used, keeping them loaded would use up memory, so they should be stored in strings so that they will only be loaded when used and unloaded when unused.

2. **Size of the image**: if the image is large, like wallpapers, they will create Surfaces with large sizes, increasing memory usage. Therefore, strings are preferred to store large images. Smaller images can be loaded without a significant use of memory.

The final choices of the data types can be seen in the section Organization of Global Variables.

G. Flowcharts

The following few pages contains the flowcharts and diagrams showing the operation of several system modules. They may later be referenced by other sections, like program documentation.

Table of Contents

Figure 1: `The Becoming of a Mathematician.py`

This is the main program, executed when the game is started up.

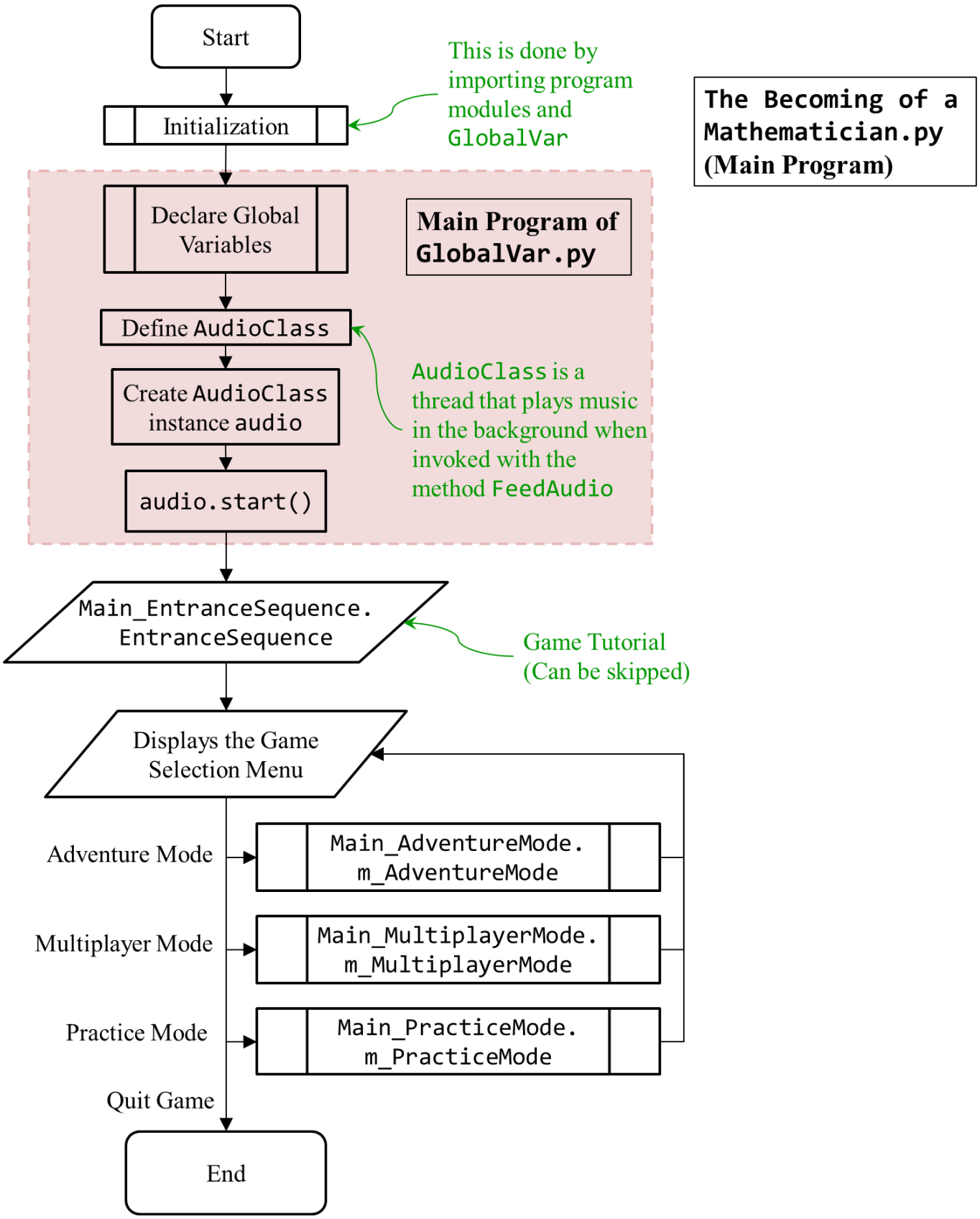


Figure 2: `Main_AdventureMode.m_AdventureMode`

This is Adventure Mode. The following shows everything starting from entering Adventure Mode to leaving.

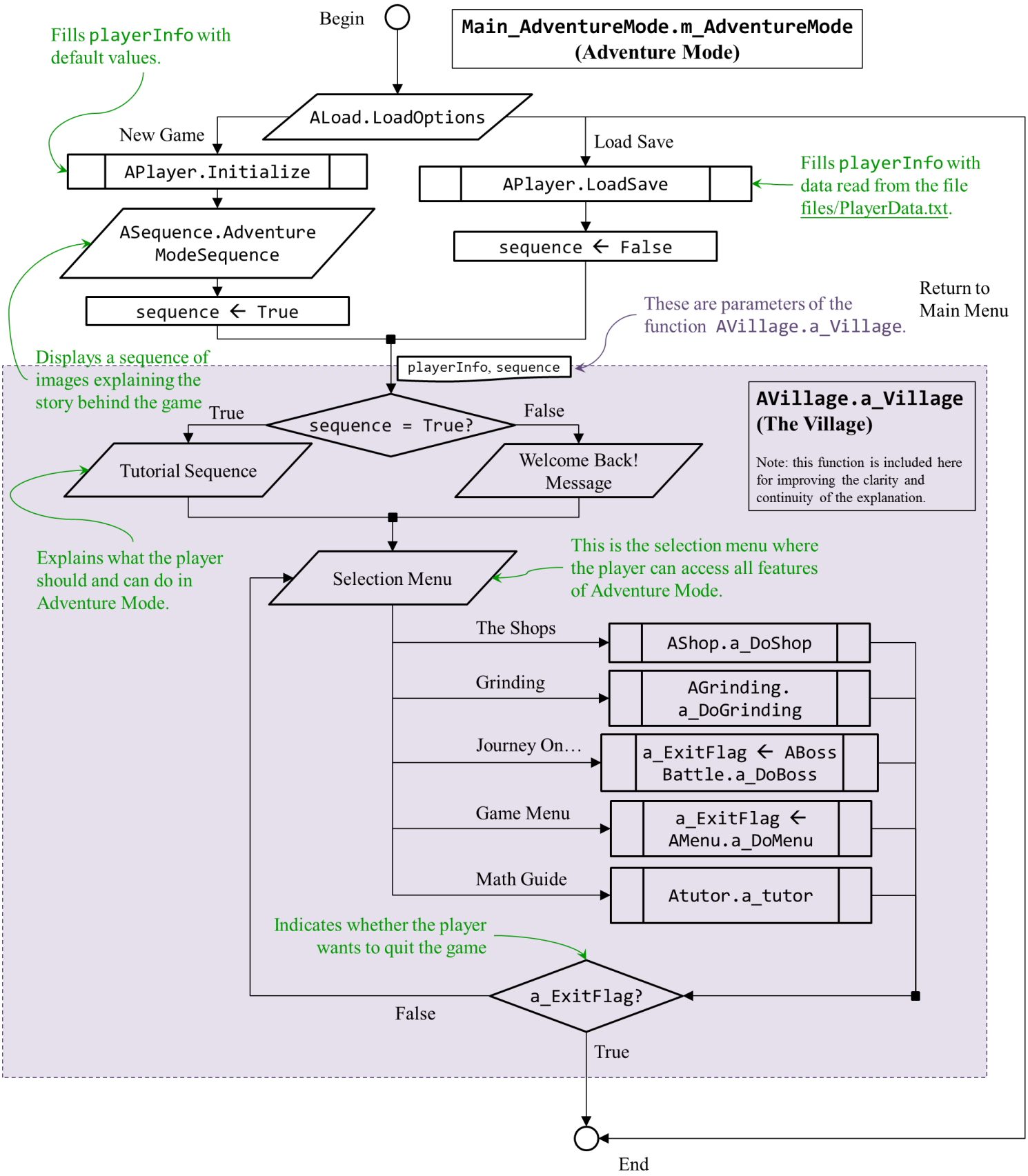


Figure 3: `ABossBattle.a DoBoss`

This is the "Journey On…" in Adventure Mode, where the player fights Mathematicians. "Grinding" varies only slightly from this, so its flowchart will not be included in this report.

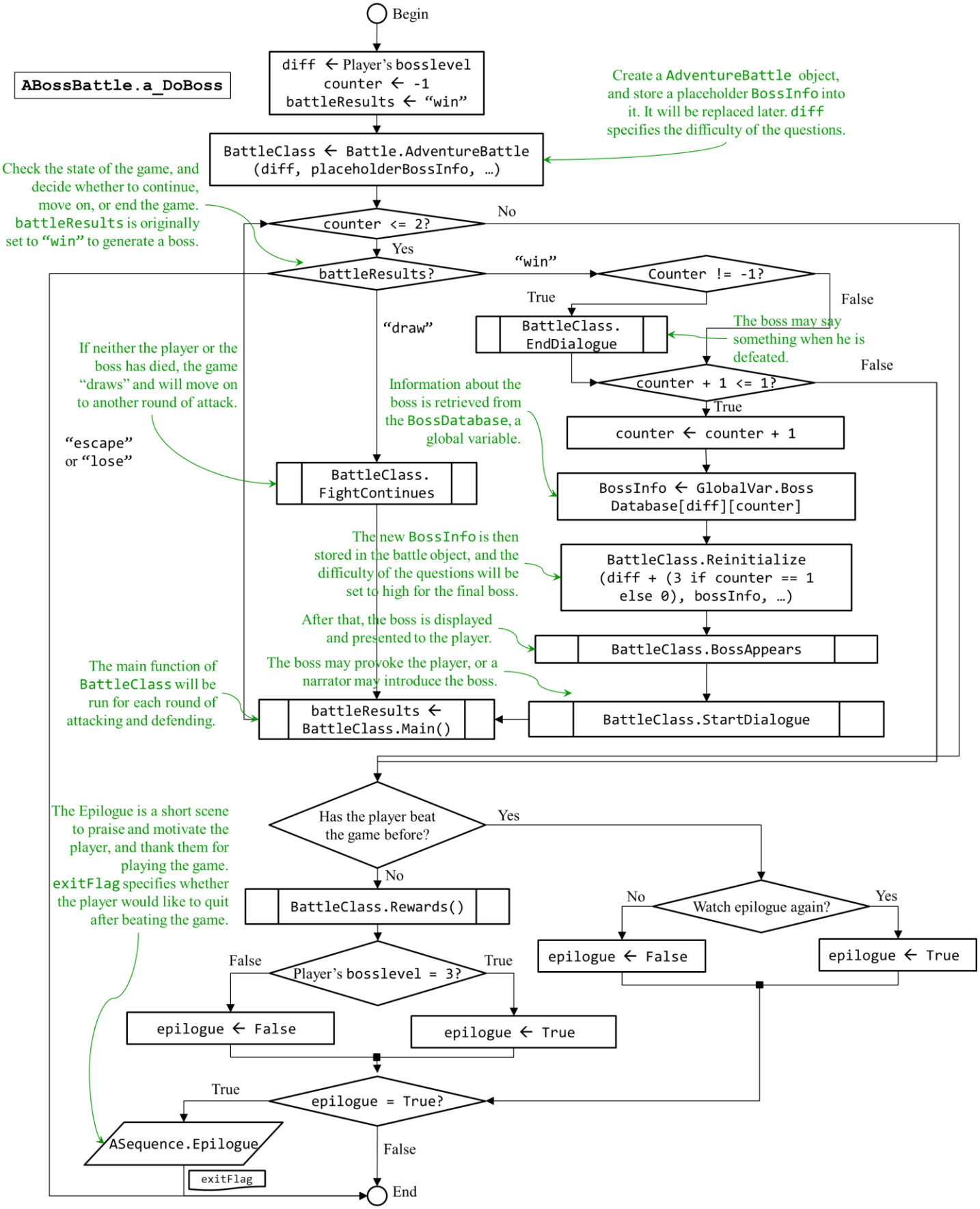


Figure 4: `AdventureBattle.Main`

The following shows the processes inside a Battle, in particular a battle in Adventure Mode.

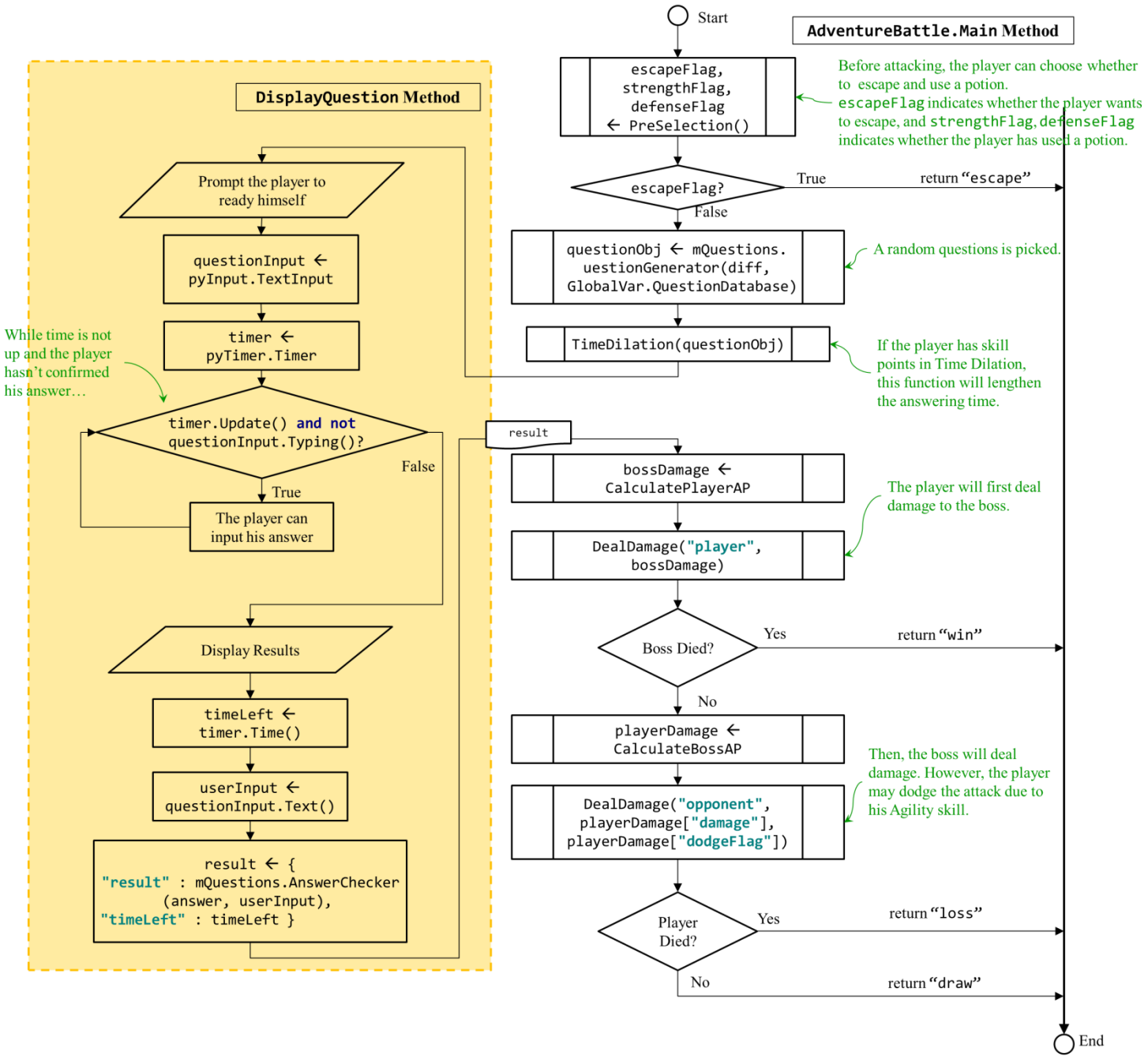


Figure 5: Online Multiplayer Battle – Timeline

This is the interaction between the client and server in Online Multiplayer Mode, from the start of connection to the end of the game.

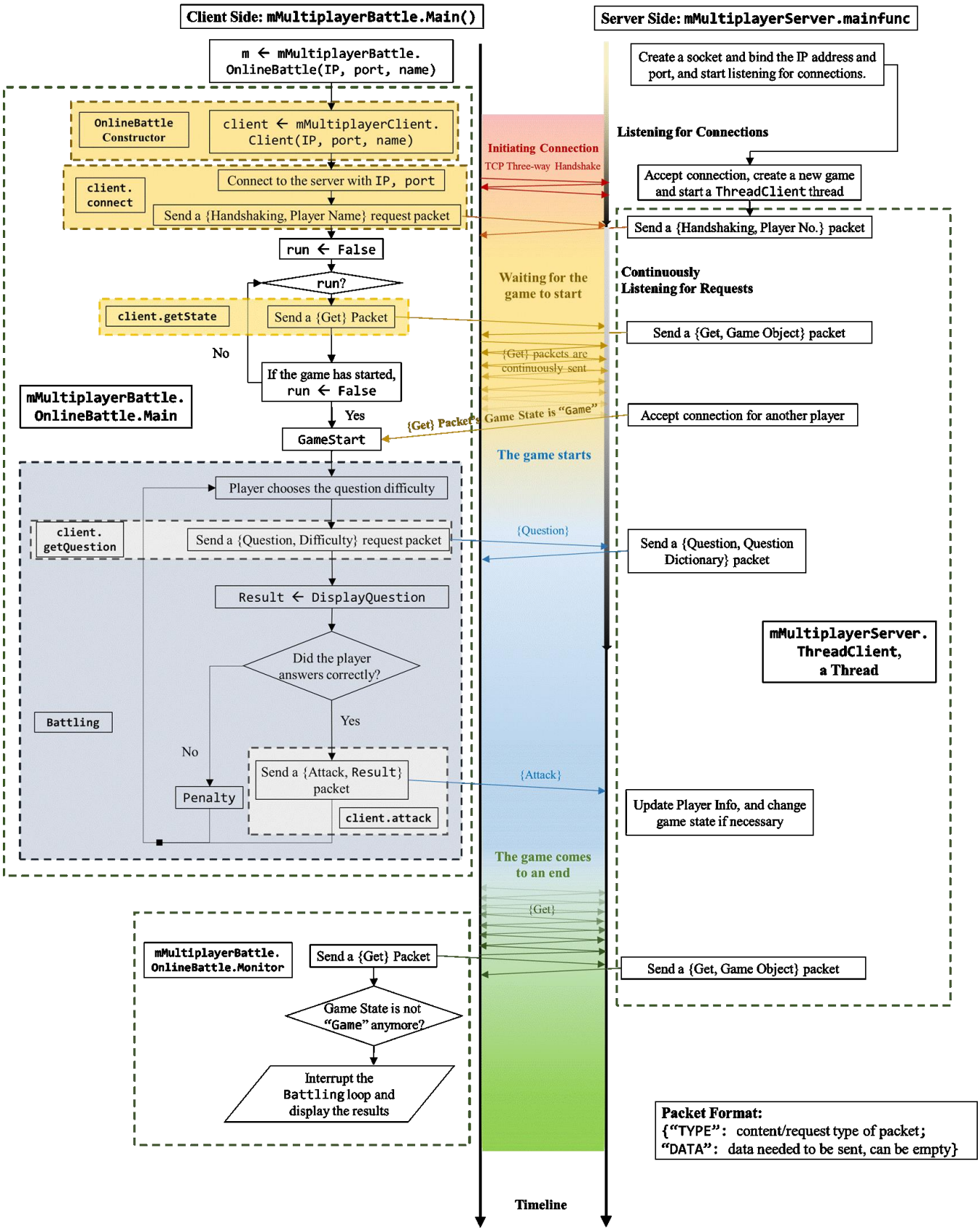


Figure 6: `mQuestions.QuestionGenerator`

This is how a question is created using the Question Generator.

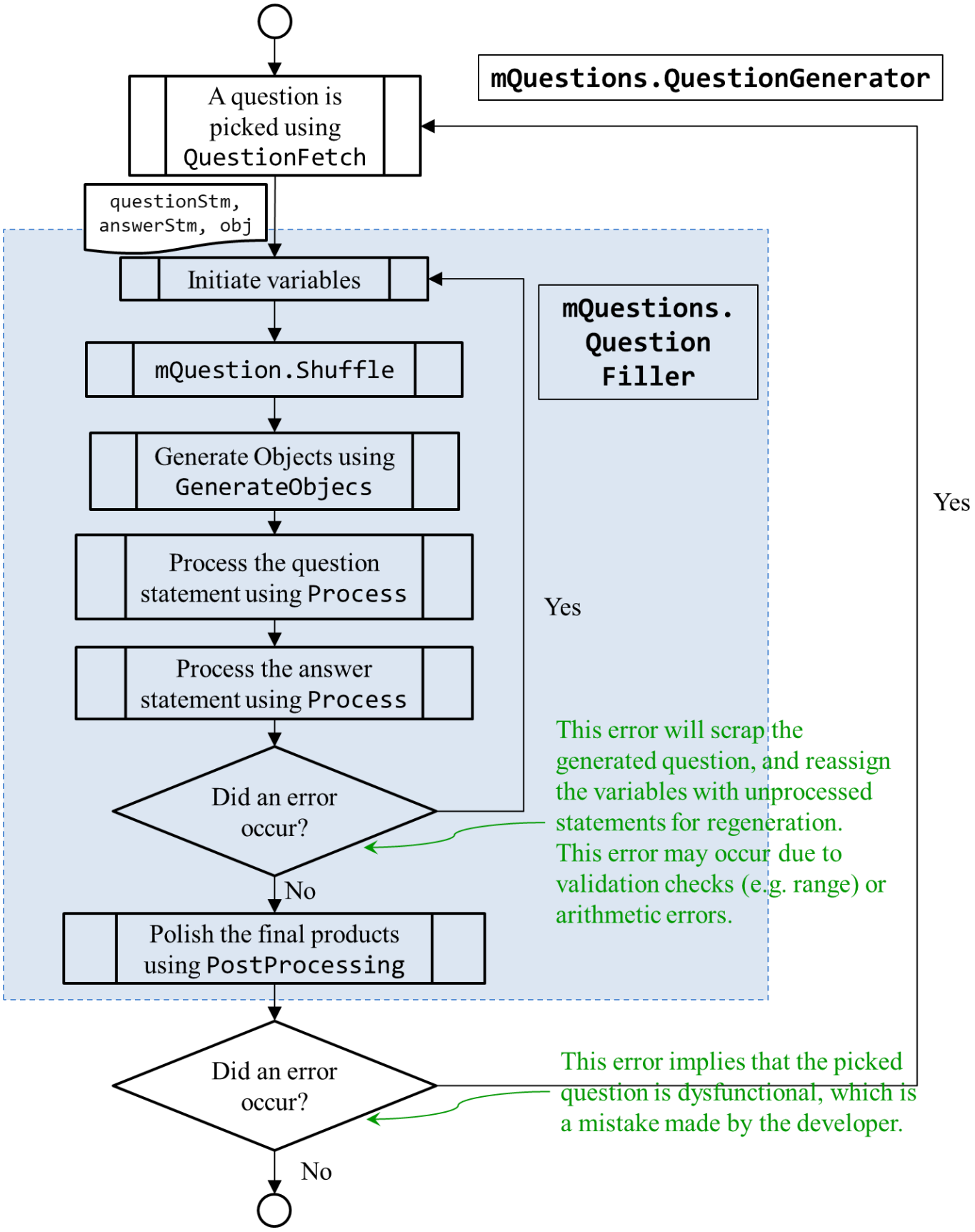
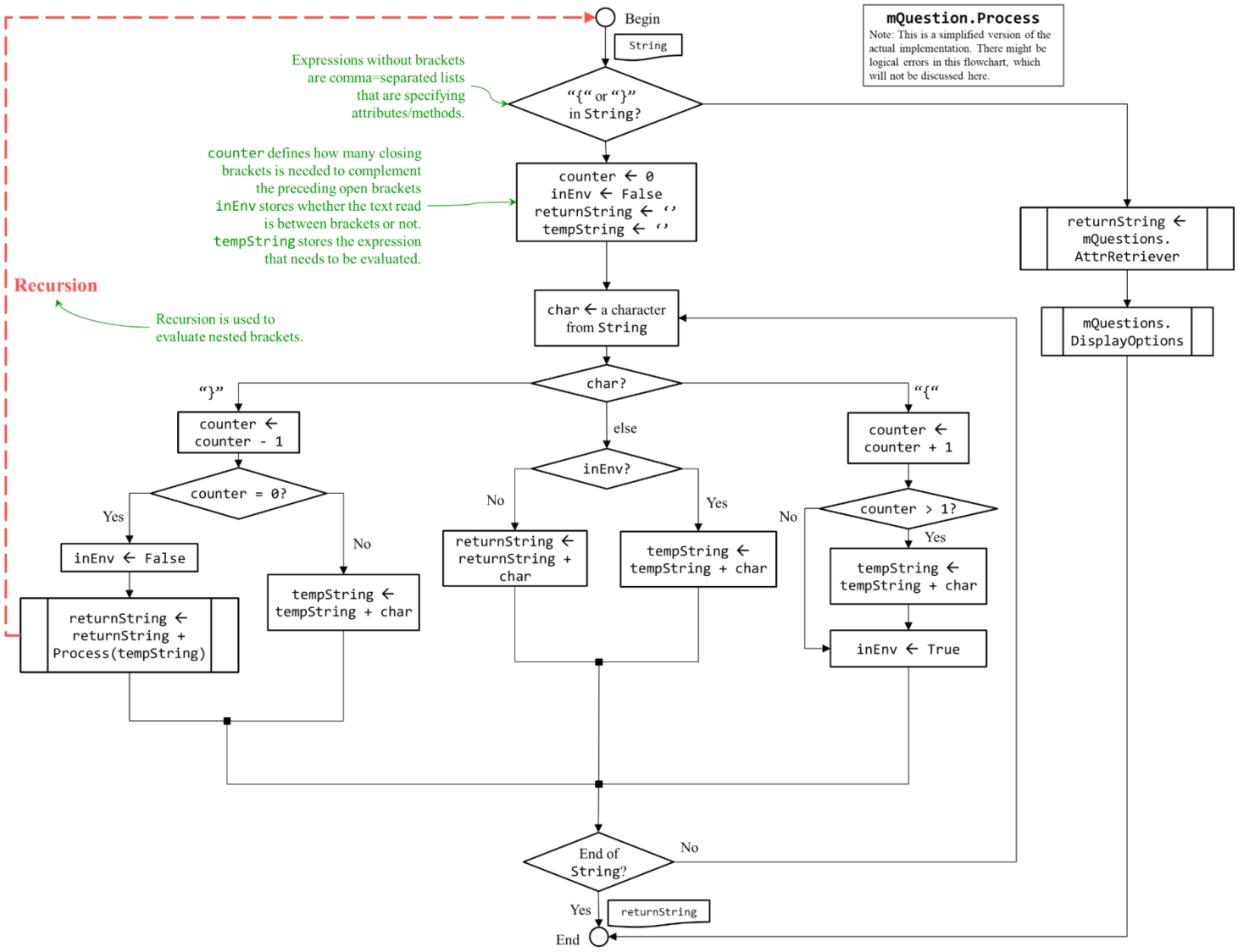


Figure 7: `mQuestions.Process`

This is how the program processes the question and answer statement in a question object, and retrieves the required attributes for it.



H. Game Manual

The following section is a user manual on how to open and play the game, along with some screenshots to present what the game looks like.

Table of Contents

1. Installation

There are two ways to open the game:

▲ **Default Profile Picture**

i. Executing the Source Code using a Python Interpreter

1. Install the latest version of Python 3. Make sure that pip (Package Installer Python) is also installed with it.
2. Install the following program libraries[9]: `pygame` (Version 2.0.0 dev 6), `simpleaudio`, `wave` and `pyaudio`.
3. Now, you may open the game "The Becoming of the Mathematician.py" using Python. To open a server, run "mMultiplayerServer.py".

ii. Compiled Executable File

Simply open the executable file named "The Becoming of the Mathematician.exe" on Windows.

Recommended Working Enviroment:

- Python 3.8
- Windows 7 or 10
- *At least* 100 MB of memory

2. Game Settings

To change the game resolution and disable/enable background music, open Game Settings/GameSettings.txt and edit the corresponding game settings. You may also change your profile picture: simply replace Game Settings/ProfilePicture.png with something else, renaming it as "ProfilePicture", and make sure the file is either in png or jpeg format.

```
# The resolution of the game. The recommend aspect ratio is 4:3
# Only expand the window horizontally out: thinner windows might cause text t
go out of bounds
# Windows that are too small might cause the program to crash (due to font si
reducing to 0 or negative values)
# Default: (1200, 800)
# where 1200 is the width in pixels and 800 is the height

Resolution: (1200, 800)

# Whether your computer displays Unicode characters
# If no, type "False" or "0"
# Default: True

Unicode: True

# Whether you want music to be played or not.
# There will still be sound effects like button clicking.

Music: True
```

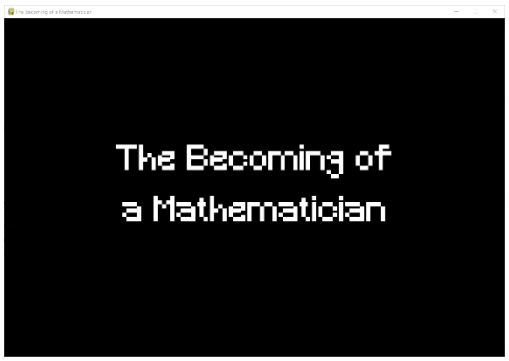
▲ **The game settings file, *Game Settings/GameSettings.txt*.**

[9] For `pyaudio`, find the appropriate .whl (wheel) version from *https://www.lfd.uci.edu/~gohlke/pythonlibs/#pyaudio*. To install these packages, open the Command Prompt, type in "`pip install {package name}`". For `pyaudio`, change the directory to the one with the .whl file, and for `{package name}` type in the filename of the .whl file.

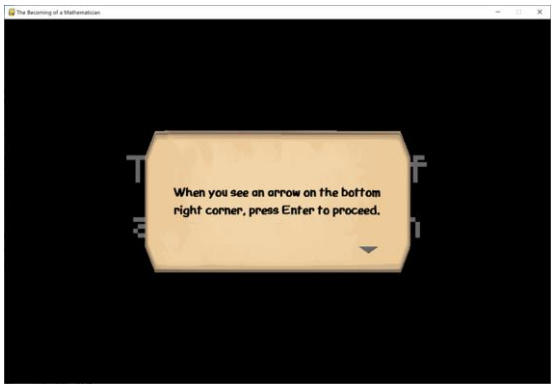
3. *Start-up*[10]

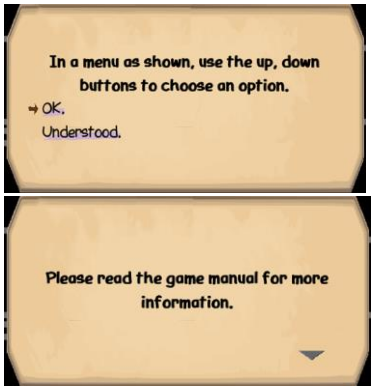
The game starts up with a welcoming screen.

Then, a short tutorial will be played to teach the player how to navigate through game menus using the keyboard and the mouse. The tutorial can be skipped in the future by pressing ESC.



▲ The start-up screen.

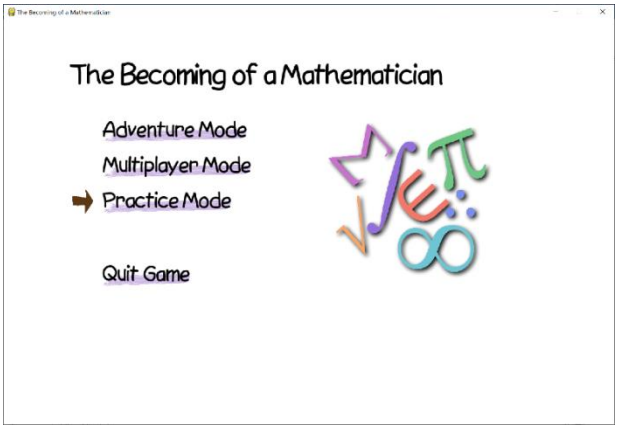




▲ A short tutorial on game controls.

4. *Main Menu*

This is the main menu of the game. The player may choose to play *Adventure Mode*, *Multiplayer Mode* or *Practice Mode* here.



▲ The main menu.

A Note on Game Controls

Keyboard: Using the arrow buttons (↑, ↓), you can move the brown arrow in the menu so that it goes up and down the menu. When the arrow stops at your desired choice, press Enter to select it.

Mouse: When you place your curser over a choice, its purple label will turn yellow, and the arrow will point at it. Click to select it.

▲ Before and after hovering over a choice.

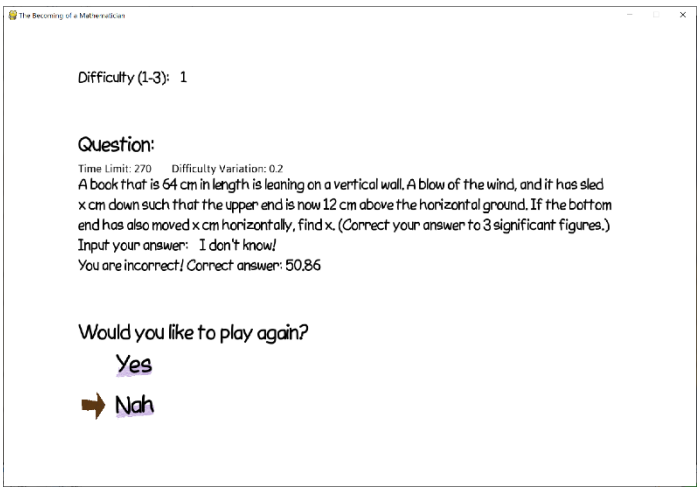
[10] Start-up takes a while (~ 30 seconds) for some reason. This is an improvement.

5. *Practice Mode*

The player can directly access the question generator in Practice Mode.

The player will first select a difficulty (1 to 3), each one corresponds to a type of question. Refer to *1C Area of Learning* for more information.

Then, a question will pop up. After answering the question, your answer will be checked. You can then play again or leave.



▲ **Practice Mode.**

Format of Answers

There are a few rules that should be strictly followed when reading and answering questions, or else you might misunderstand the question and the program might misinterpret your answer, causing your answer to be treated as incorrect. They are as follows:

1. **Without parenthesis, * and / takes the sequence of digits between it and another operator.** * and / are the multiplication and division operators respectively. This rule states that the expression `a/b+c` means $\frac{a}{b}+c$, but not $\frac{a}{b+c}$, and `a*b+c` means $ab+c$, but not $a(b+c)$.

2. **Exact answers must be given.** That means to write your answer in fractions or exact decimals (approximate decimals, like recurring decimals, will not work). For example, the answer of the question "Solve $5x = -3$." is $x = -3/5$. The only acceptable answers are `-0.6`, `-3/5`. The answer to "Solve $3x = -5$." can only be `-5/3`, neither `1.66666`… nor `1.6666666666666666666666667`.

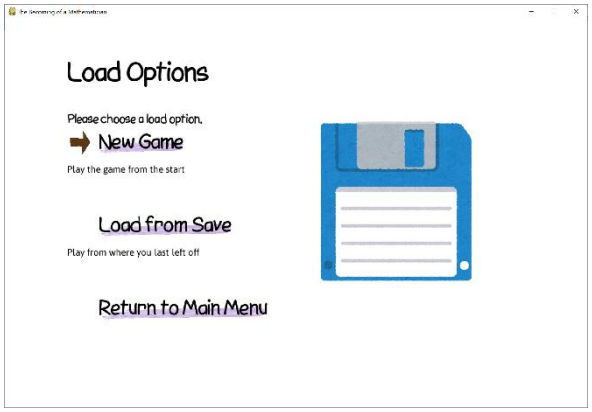
3. **Multiple answers are separated using commas.** Questions like quadratic equations and sequences often involve multiple answers, so the answers are separated by commas. The answers to the question "Solve $x^2 - x - 2$." are represented as **"`2,-1`"**, **"`2, -1`"** or **"`-1, 2`"**. Again, answers must be exact.

6. *Adventure Mode*

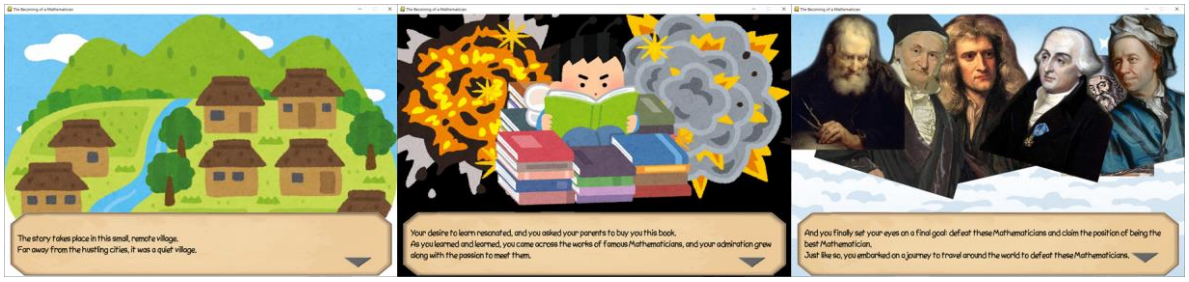
i. *Loading Options*

The game progress in Adventure Mode can be saved, so the player first chooses where to start the game from in the Load Options. The plaer may start a new game with zero progress, or continue his previous game by choosing "Load from Save", **given that the player has saved the game before**.

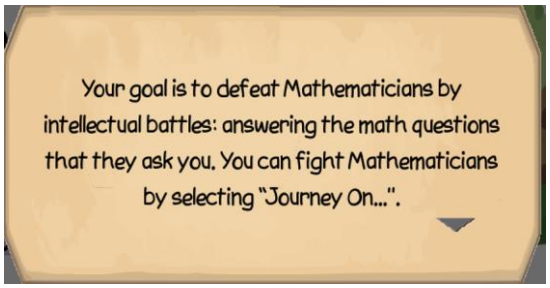
Starting the new game will initiate a short storyboard to introduce the player to the story behind the game, and a tutorial to teach what the player's goal is in Adventure Mode.



▲ Load Options for Adventure Mode.



▲ A short story about the background of the game, the "Entrance Sequence".



▲ A short tutorial on Adventure Mode.

After the tutorial, the game officially starts. In the village, the player may choose “The Shops”, “Grinding”, “Journey On…”, “Math Guide” and “Game Menu”. In the following sections, these features will be introduced one by one.

▲ The Village.

ii. *The Shops*

▲ Armour and Weapons Shop

▲ Potion Shop

▲ Skill Shop

The three shops are “Armor and Weapon Shop”, “Potion Shop” and “Skill Shop”.

Purchasing in The Shops: an Example

Suppose we want to buy 8 Potions of Regeneration in the Potion Shop with 10000 dollars.

We first select the product that we want by hovering over it and selecting it. This can be done either by using the keyboard (arrow keys and Enter), or the mouse (hover and click).

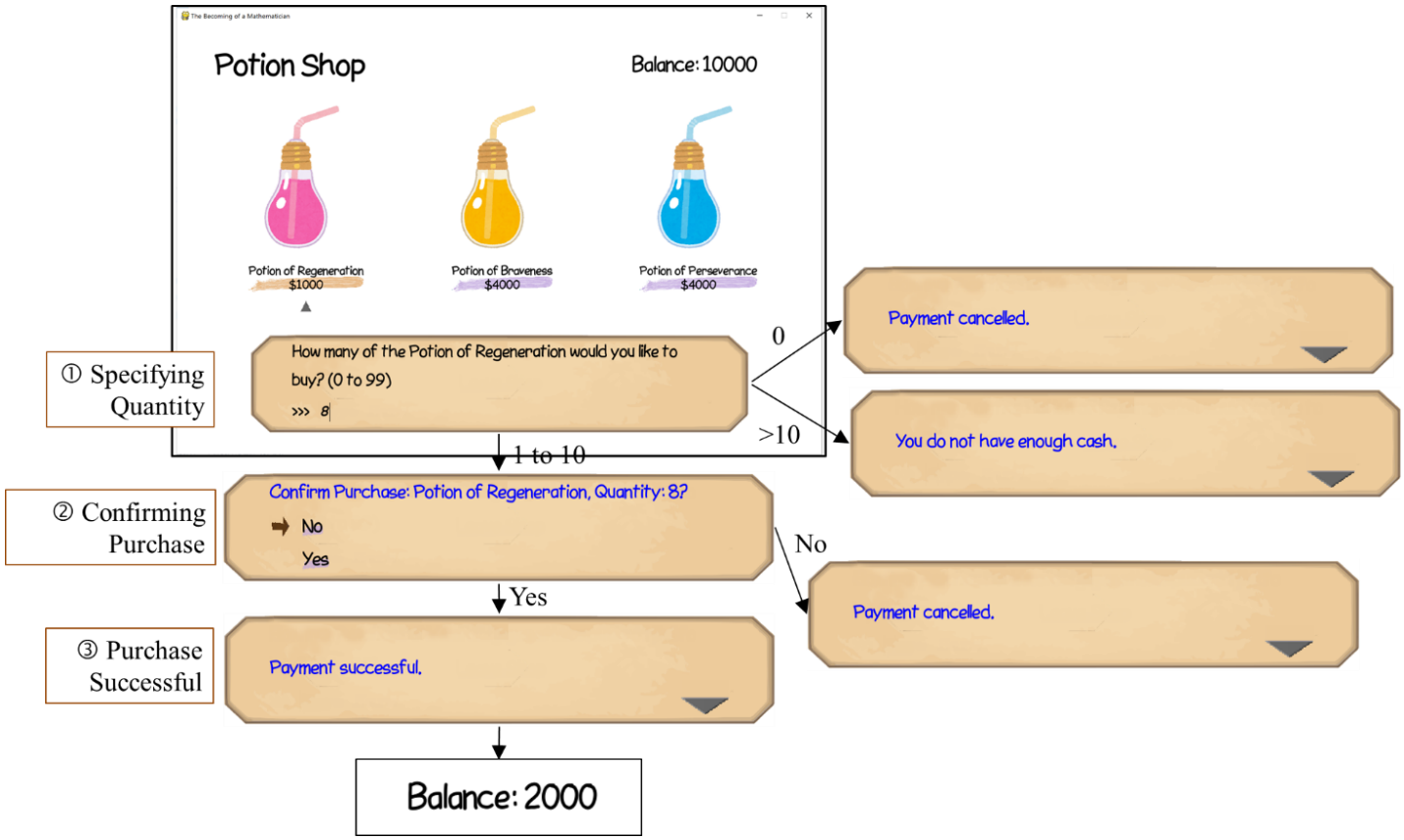
▲ The Shops' main menu.

Then, it will ask the user for the quantity (Step 1). Here,

- typing 0 will cancel the purchase,
- 1 to 10 to buy 1 to 10 potions, while
- any number larger than 10 will display the "You do not have enough cash" message.
- Furthermore, when text that is not a number is entered, the program will do nothing and wait for you to delete it and type an appropriate response.

▲ Entering the Potion Shop with 10000 dollars.

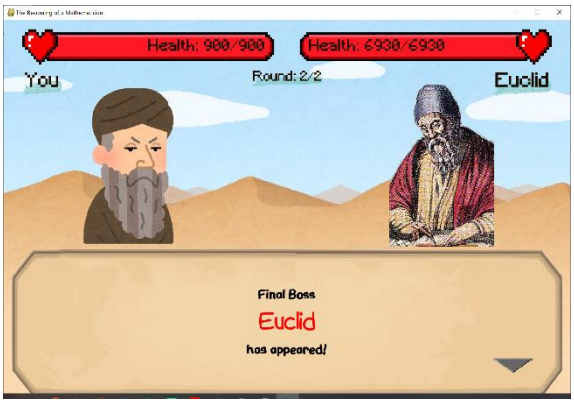
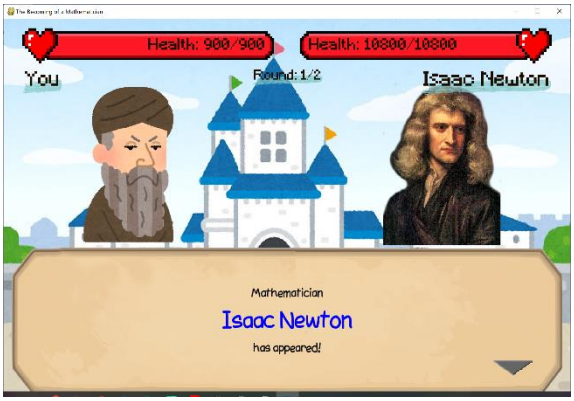
After typing in "8", you will be asked to confirm your purchase (Step 2). After confirmation (Step 3), you will have successfully bought 8 potions, and your balance will be deducted accordingly.



▲ Purchasing in the Shops.

iii. *Journey On…*

"Journey On…" is where the player can progress into the game by fighting mathematicians. There are a total of 6 Mathematicians, 2 Mathematicians per level and a total of three levels.

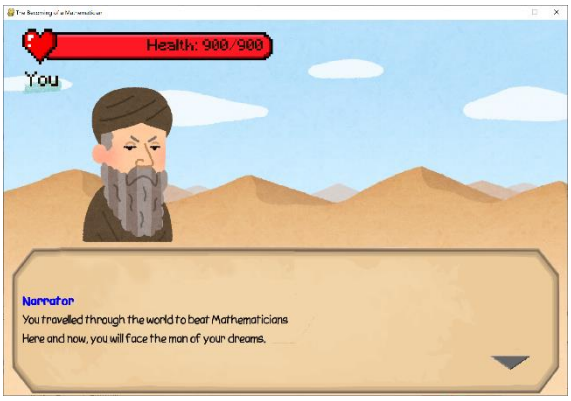
		Round	
		1	**2**
Level	**1**		
	2		
	3		

▲ The six Mathematicians in the game.

There will be two rounds in each fight. The rest of the section will explain, step by step, how the fight is carried out.

(1) Mathematician Appearance

The Mathematicians will first appear, and their name will be shown, as seen in the above table. Then, the narrator will talk about who they are and what contributions did they make in the advancement of Mathematics. Lastly, the Mathematicians may provoke the player before actually beginning the fight.



▲ **Journey On...**

▲ **The Narrator introduces the boss.**

▲ **The boss may provoke or talk to the player.**

(2) Before Attacking

A selection menu will then pop up. The player can escape (i.e. giving up the fight and returning to the village) or use a potion (only one). After the player is ready and selects “Proceed to Question”, a reminder telling the player to get ready will pop up.

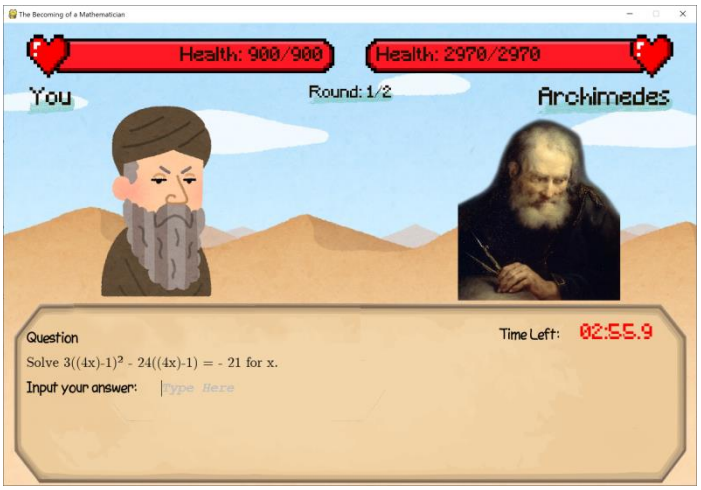
▲ **A selection menu to escape, use potions or proceed to attack.**

▲ **The player may choose one and only one potion.**

▲ **A reminder telling the player to get ready for the question.**

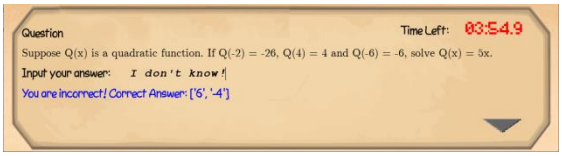
(3) Attacking

To attack, the player has to answer a math question within the time limit. The final amount of damage depends on several factors, like answering time, equipment, skills and potions.



▲ **Answering the question, $3(4x-1)^2 - 24(4x-1) = -21$.**

▲ **Answering a question correctly.**



▲ **Answering a question incorrectly.**

(4) After the Attack

The player will first attack the Mathematician. If the Mathematician survives, he will attack the player. If the player dies, then the fight ends, and otherwise it will continue onto another round of answer and attack.

▲ **The fight continues if neither dies.**

▲ **Attacking the Mathematician.**

▲ **Being attacked by the Mathematician.**

(5) Rewards

When the Mathematician dies, rewards will be given to the player. When the player selects "Journey On..." or "Grinding" again, the next level of questions will be asked, and the next set of Mathematicians will appear.

▲ **Rewards**

iv. *Grinding*

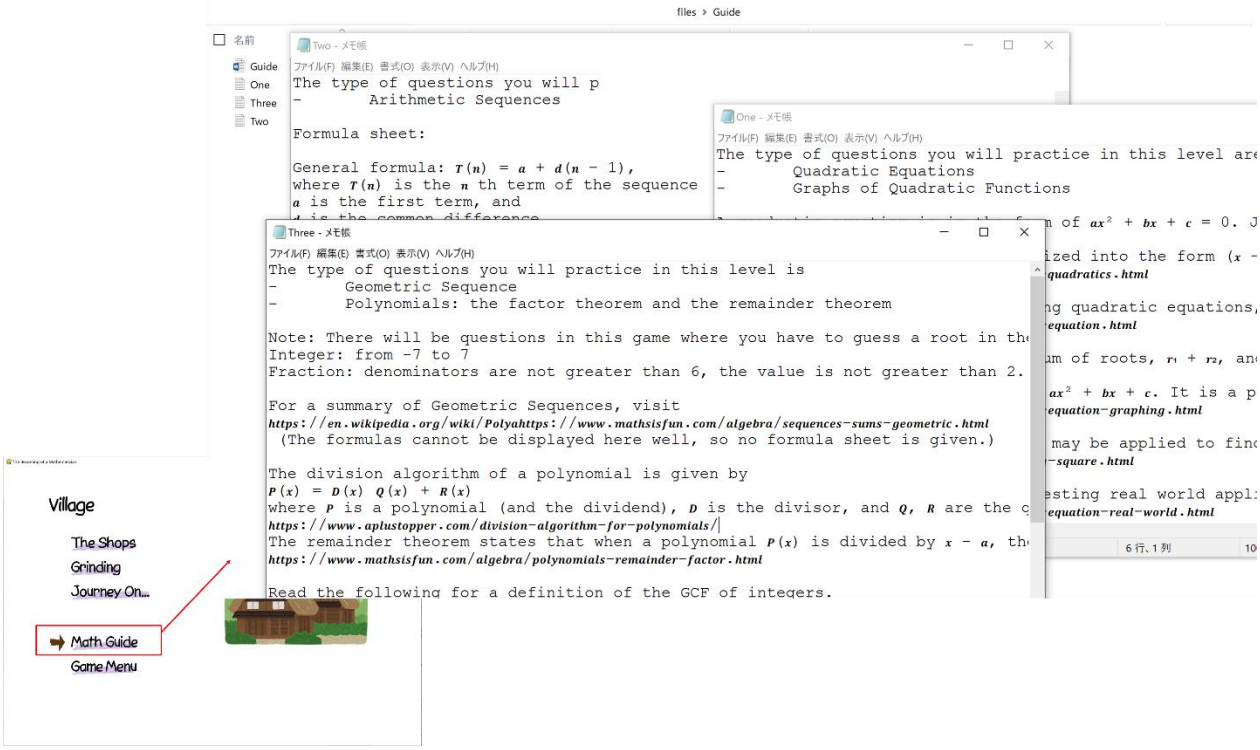
The flow for Grinding is exactly the same as “Journey On...”, so we will be omitting its details.

▲ Grinding.

▲ Brawling with a Monkey.

v. *Math Guide*

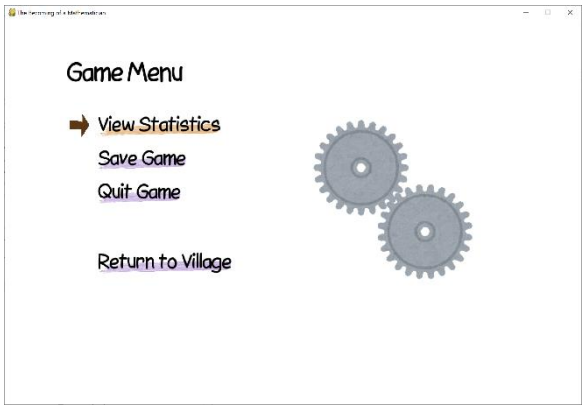
When the Math Guide is selected, a text file will pop up, informating the player about type of question in each level and some useful online resources. After beating the game, the folder containing all previous guides will be opened instead.



▲ Math Guide.

vi. *Game Menu*

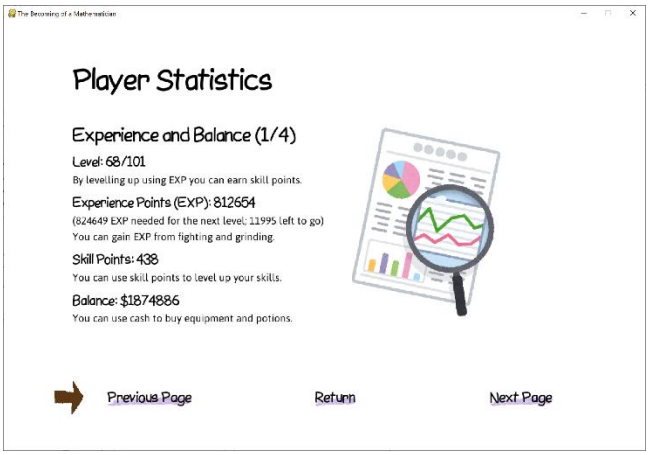
The Game Menu allows the player to view their progress in the game, or save/quit the game.

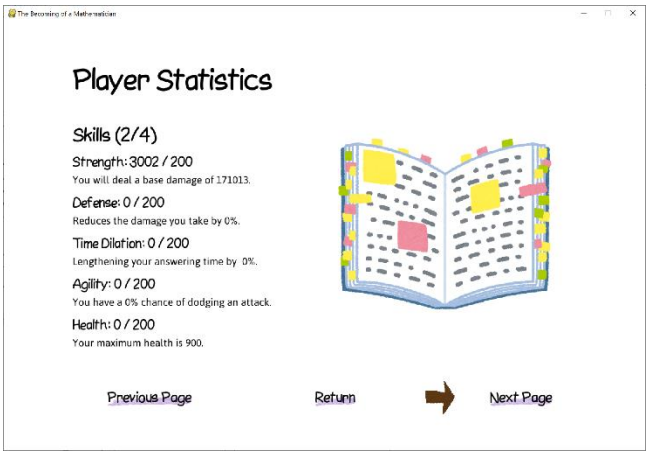


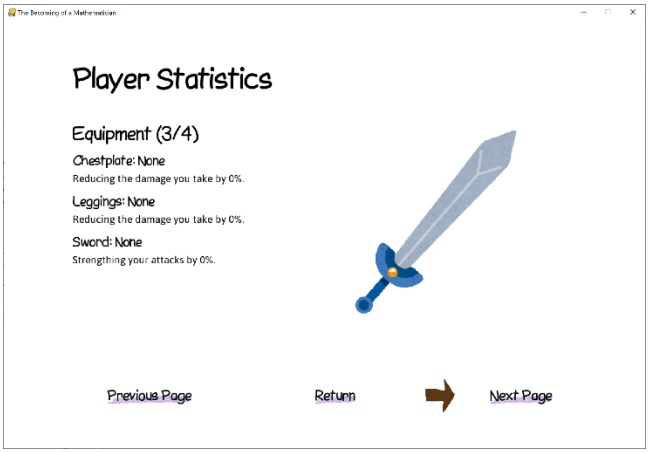
▲ **The game menu.**

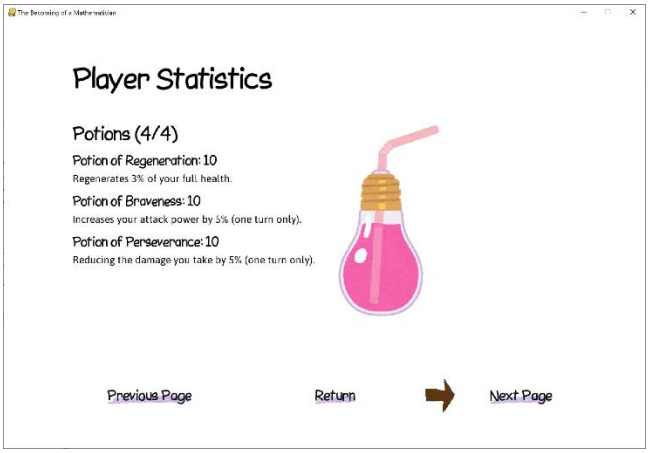
(1) Game Statistics

The player may view their progress in the game. There are 4 pages in the game statistics: Experience and Balance, Skills, Equipment and Potions.







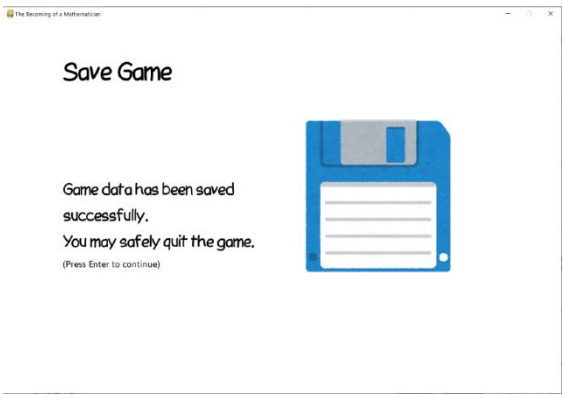


▲ **The 4 pages in View Statistics.**

(2) Save Game

You may also save your game to carry on at a later time, since it is not intended for the player to complete the game in one game session.

Besides, while rare, it is possible for the game to crash suddenly due to its high memory usage. Game saving is not automatic, so it is a good practice to occasionally save the game[11].

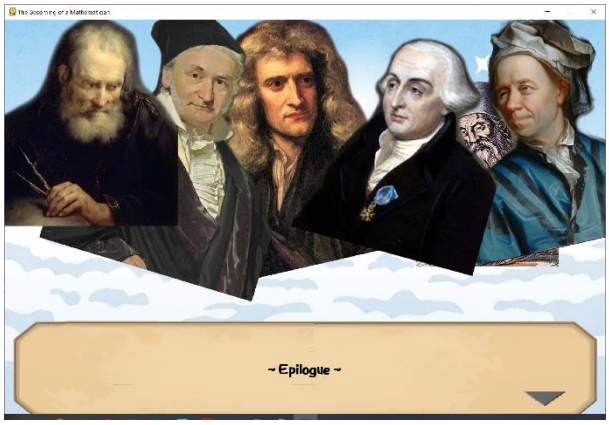


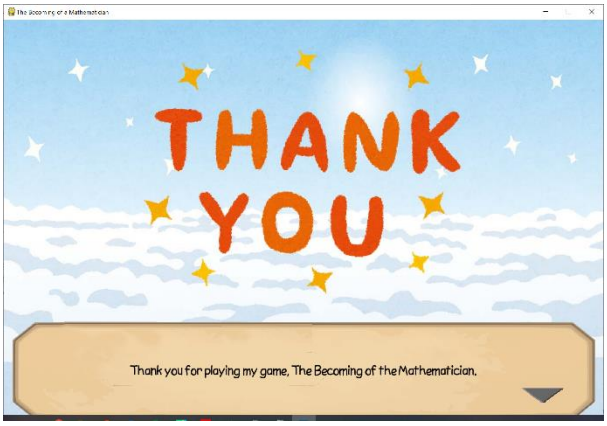
▲ **Save Game.**

[11] This is an improvement.

vii. *Epilogue*

After beating the last Mathematician, the game will automatically enter the Epilogue. It is a speech of gratitude to the players for playing the game, and an encouragement to their future endeavors in Mathematics.





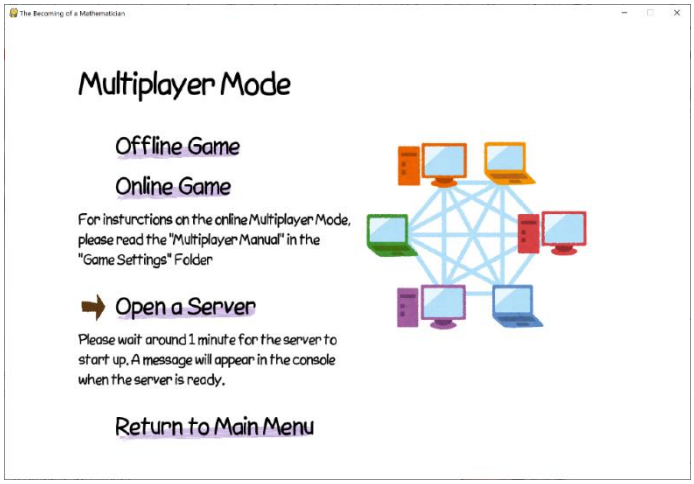
▲ Screenshots of the Epilogue.

7. *Multiplayer Mode*

Multiplayer Mode involves the interaction between two players. There are two Multiplayer Modes: *Offline Multiplayer Mode*, and *Online Multiplayer Mode*.

i. *Offline Multiplayer Mode*

In Offline Multiplayer Mode, two players take turns to attack until either side dies. The players can choose what question category they'd like to answer. (Different caregories deal different magnitudes of damage.)



▲ Multiplayer Mode.

▲ Changing Turns.

▲ In Multiplayer Modes, the players are allowed to choose a difficulty.

The end of the game. ▼

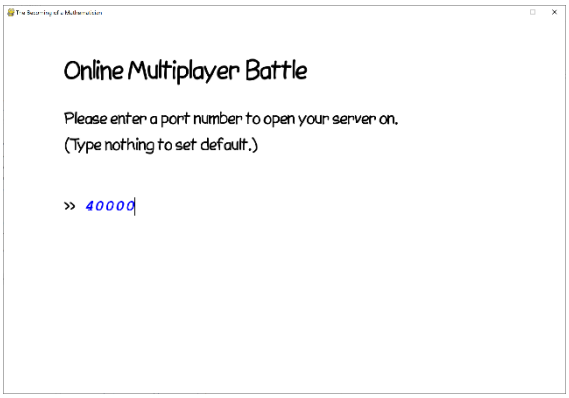
ii. *Online Multiplayer Mode*

The Online Multiplayer Mode involves opening a server and connecting to it. The server accepts connection from computers, and it pairs up two connections to form a game. The following section will describe how to initiate an Online Multiplayer Mode game.

(1) Opening a Server

There are two ways to open a server. One is to run `mMultiplayerServer.py` and type in the connecting details, including the IP address and port. Another one is to open it within the game. The "Open a Server" option in the Multiplayer Mode menu is used to open a server.

The default port number is 40000. It can be changed to other ports if 40000 is not available.



▲ Opening a server from inside the game.

After opening a server, a message will pop up in the console. In the message, the set of numbers "192.168.1.74:40000" is called a socket address. The first set of numbers "192.168.1.74" is the IP address[12], while the number "40000" after the colon is the port number.

```
libpng warning: iCCP: known incorrect sRGB profile
Server:  Up n' runnin'!
Server:  Connect to this server by:  192.168.1.74:40000
```

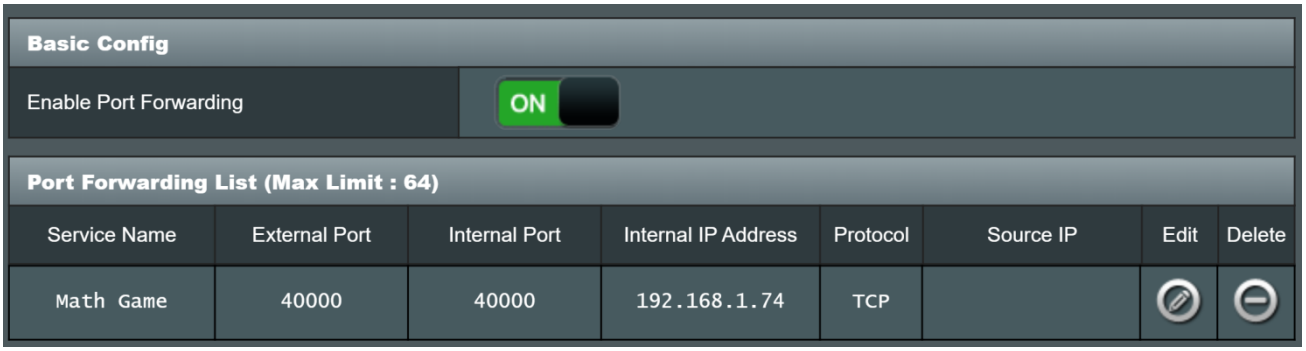
▲ After opening a server, the following message will pop up in the console. The sequence of number given is the socket, which is used to connect to the server using "Manual Connection".

Port Forwarding and Firewalls

If you expect **computers from outside your network to connect to your server**, there are two things that you have to configure.

The first one is Port Forwarding: clients must connect to your server with your Physical IP, so Port Forwarding redirects a communication request through one of your router's port and forwards it to your computer's port. For example, if you opened your server on your computer with the private IP 192.168.1.74 and at port 40000, and you would like other computers to also connect through port 40000 of your router, you have to route port 40000 to port 40000 of 192.168.1.74, as in the following image:

[12] Specifically, this is the private IP address.



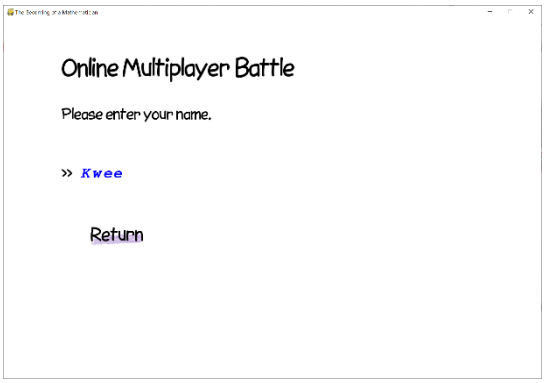
▲ Port Forwarding.

The second one is to allow your Server through the Firewall. Open the game or the server at least once, and open **Choose Start → Control Panel → System and Security → Allow a Program through Windows Firewall**. Find Python (if you are running the python script with Python), "The Becoming of the Mathematician.exe" or "mMultiplayerServer.exe" (if you are running the executable), and tick both Private and Public.

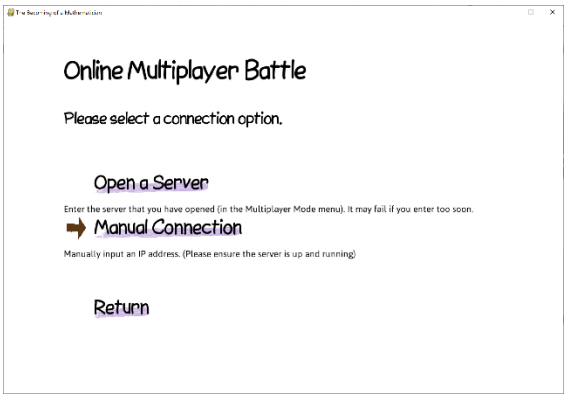
(2) Connecting to the Server

After opening the server, it is time to connect to the server. Choose "Online Multiplayer Mode", and you will be prompted to type in a username. This username will be the name shown to your opponent.

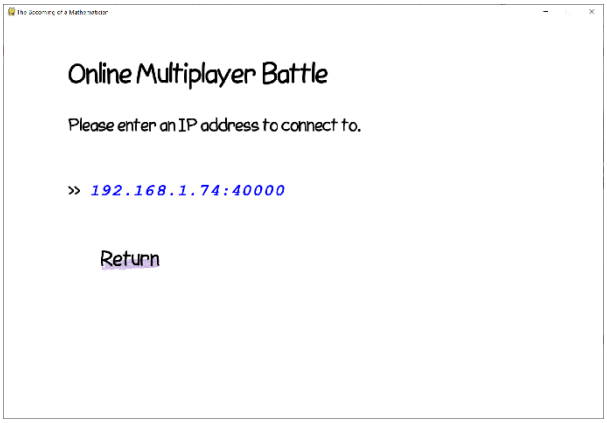
After that, you will arrive at "Connection Options", where you have two choices: "Open a Server" or "Manual Connection". The former connects to a server opened with your IP address, while the latter connects using the IP address entered by the user.



▲ Entering a username.



▲ Connection Options.

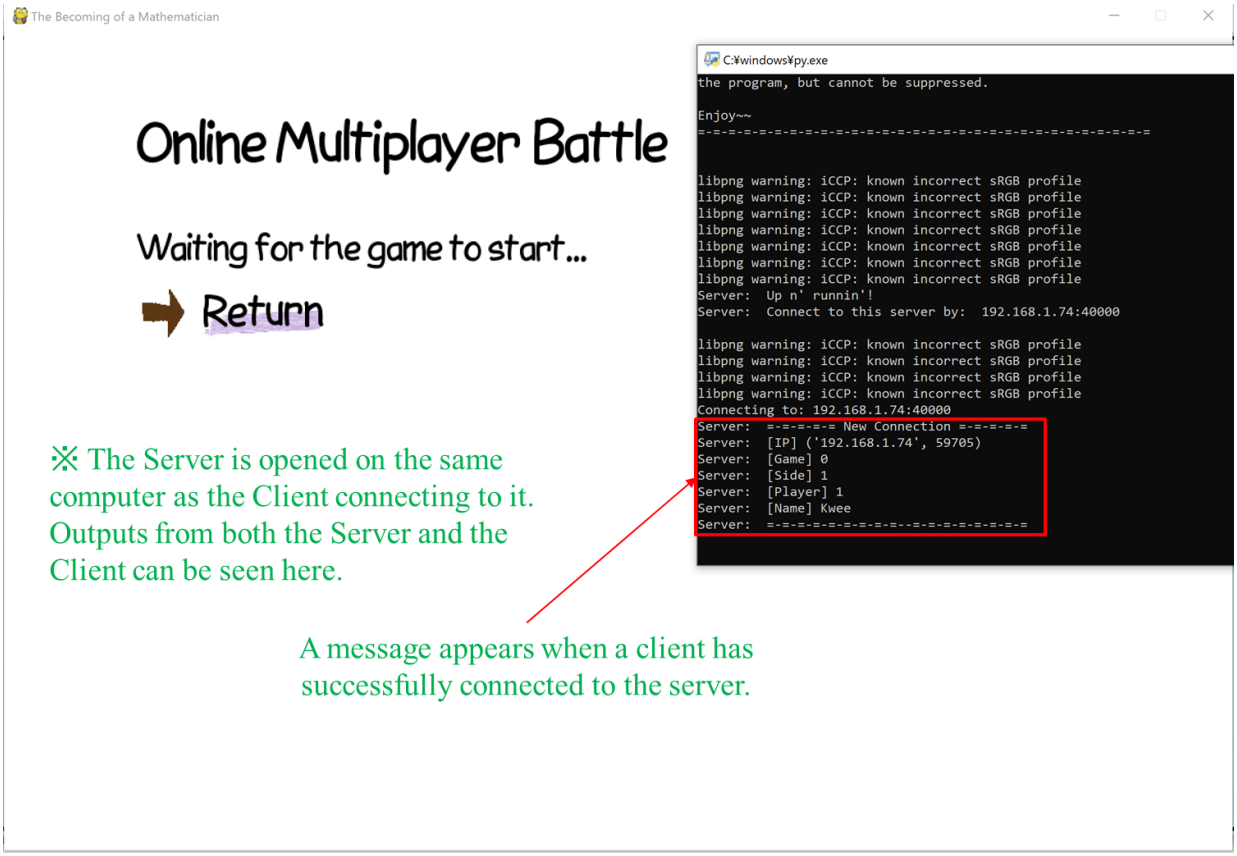


▲ Manual Connection

There are three types of connections that can be made, and each will require different connection configurations[13]:

1. **Joining the server opened on the same computer**: Simply select "Open a Server". Make sure you have opened the server by checking if there is a message in the console.
2. **Connections from a computer on the same network as the server**: in this case, choose "Manual Connection", and directly enter the socket given by the server's message to connect to the server.
3. **Connections form a computer on a different network**: In this case, the physical IP should be used. The physical IP can be found by Googling "My Physical IP Address". Connect to the server via the port of the router where port forwarding is done on.

If you have connected successfully, you will see the "Waiting for the game to start" screen.



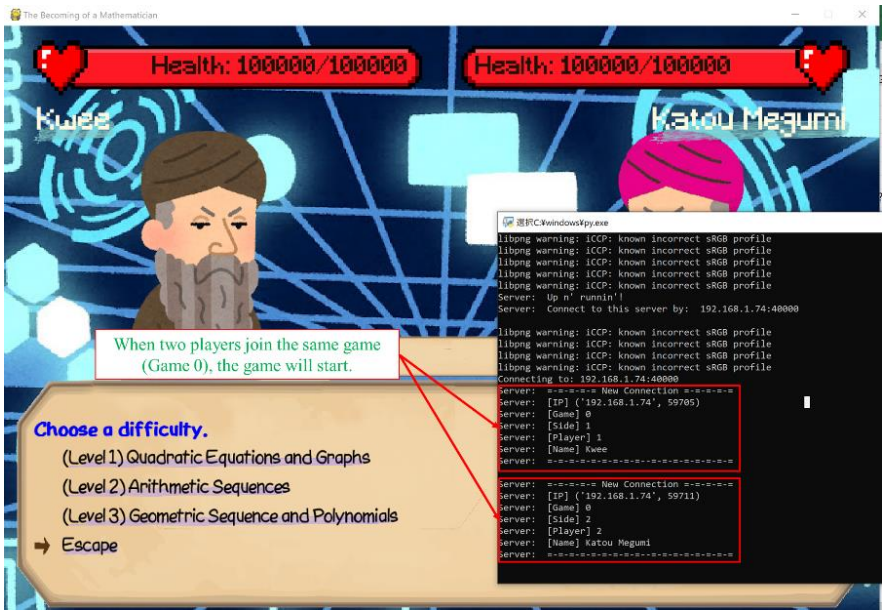
▲ "Waiting for the game to start" screen.

[13] Just to repeat, the socket format used in this program is 192.168.1.74 (IP Address) : 40000 (Port No.), where the IP address and Port Number is separated by a colon.

(3) Starting the Game

When two players connect to the same server, the server will tell both players that the game is going to start.

The player can then start attacking at any time. First, choose a difficulty (Step (1)). Then, the player will have to wait three seconds before the question is shown (Step (2)). The players can then answer the question, and if he answers correctly, an attack will be immediately made (Step (3)). Otherwise, the player will receive a 10 second penalty to prevent him from shuffling through the questions to get one he likes (Step (4)).



▲ **The "Difficulty Selection" screen.**

▲ **The question will be shown 3 seconds after selection.**

▲ **An attack is made. The status bar above the text box shows the latest attack.**

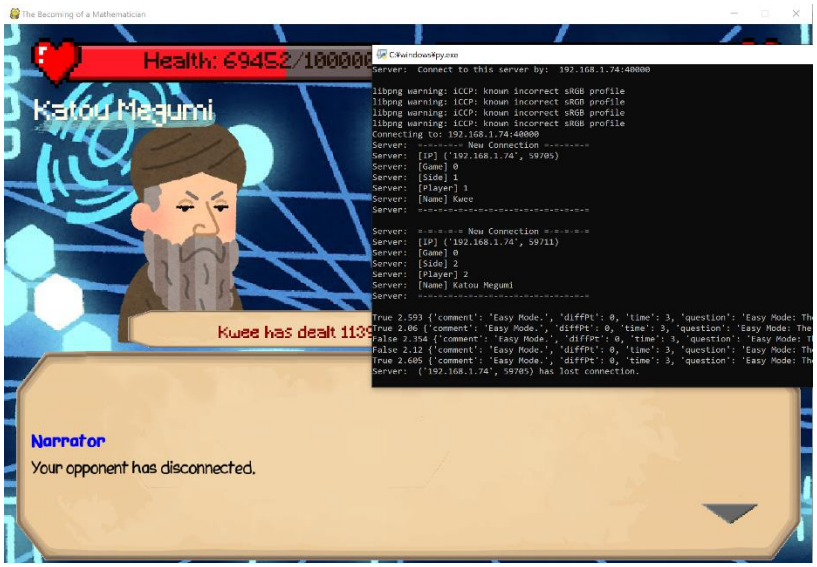
▲ **A 10-second penalty is given for a wrong answer.**

(4) Game Endings

The game may end in three ways: Winning, Losing or Disconnection. A disconnection may be caused by your opponent leaving, or by a connection failure.

▲ Winning

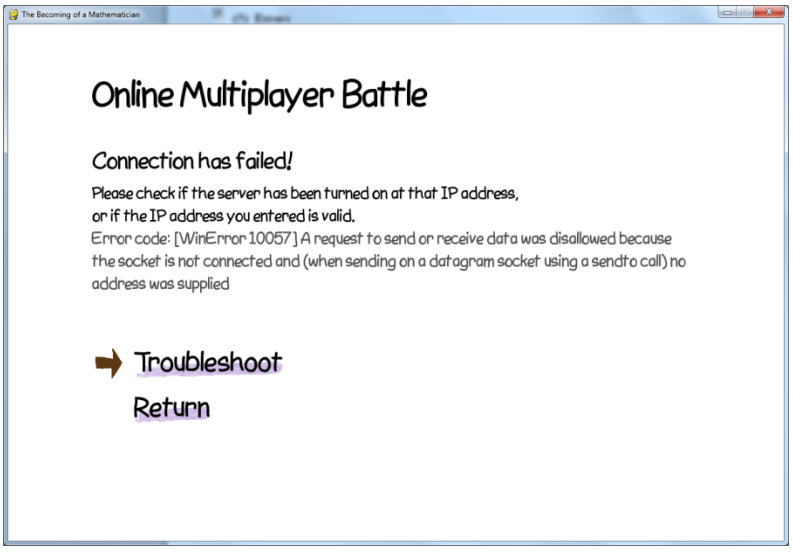
▲ Losing



▲ Disconnection

(5) Errors

Connection errors may happen when you connect the server, or when the game is ongoing. When an error happens, a "Connection Failed" error screen may appear with an error message. "Troubleshoot" shows several common error messages and how to solve them. The following are some common errors that would happen and their solutions.

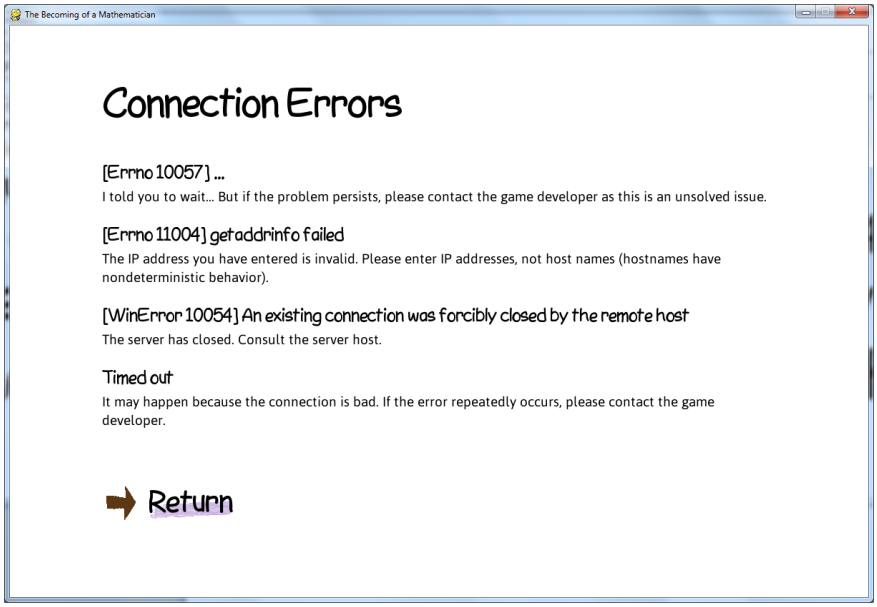


▲ "Connection has failed!" message.

[Errno 10057] A request to send or receive data was disallowed because the socket is not connected and (when sending on a datagram socket using sendto) no address was supplied.

This error means that the socket is not connected. There are several reasons for this issue:

1. The server has not been opened yet, or is still in the middle of starting up. Open a server, or wait till the server is opened.
2. There is something blocking off your connection to the server. For example, you have not port forwarded your server, or the firewall is blocking connection attempts from unknown devices.



▲ **"Troubleshoot".**

[Errno 11004] getaddrinfo failed

The IP address you have entered is invalid. Please enter IP addresses, not host names.

[Error 10054] An existing connection was forcibly closed by the remote host

The server has closed. Consult the server host.

Timed out

This error is raised because the server not giving any response to your computer for a while, after establishing a connection. It may happen because the connection is bad, or if the server has shut down.

- Invalid IP Format. Format: XX.XX.XX.XX:PortNo. Do not use hostnames.
- Invalid IP Format. Octets must lie between 0 to 255.
- Invalid Port Number.

These are all due to wrong IP address format. IP address are written as `XXX.XXX.XXX.XXX:PortNo`, where the `XXX`s are numbers from 0 to 255, and `PortNo` is a number from 1 to 65535.

~ End of Game Manual ~

2. IMPLEMENTATION

A. Hardware and Programming Language

Compiling Environment

Processor:	Intel® Core™ i7-4790 CPU @ 1.10 GHz 1.50 GHz
OS:	Windows 10 Home, 64-bit
RAM:	8.00 GB
Compiler/Interpreter:	Python 3.8

The RAM is more than enough, as the program requires low processing power.

The program will be written using **Python**.

When compared with Pascal, Python is undoubtedly better. As the language that is most used, Python has more advantages when compared with Pascal, and other languages like C, C++ and Java, in writing this game:

Advantages	Explanation
Ease of Learning	With its brilliantly simple syntax, it is undoubtedly the easiest program to learn and write with. This explains its fame as the most used programming language. This is also the main reason I have chosen Python.
Rapid Application Development	Python supports rapid application development, and it is said that it takes half as long and half the code when compared to writing in C or C++[14].
Exception Handling	Python supports exception handling by the `try, except` block which is needed in the Question Generator. This should be the easiest to learn out of all.
Functionalities and Libraries	• `eval` function, which would be very useful in implementing the Question Generator. • Game development packages like `pygame` and `arcade` with tons of support. • JSON storage for storing player data.
Concurrency	Python supports threads and multiprocessing, which is needed for things like background music and running a server.

[14] Sticky Bits: *https://blog.feabhas.com/2014/03/rapid-application-development-with-python/*. Accessed on 14th March, 2020.

B. Programming Techniques

A third of the effort put into the program is used to enhance the graphics, another third to create the Online Multiplayer Mode. The last third, is used on the Question Generation Algorithm, and is the algorithm with the most programming technique employed. This section will explain the programming techniques used throughout the program, mostly focusing on the Question Generator.

1. Question Generation Algorithm

This section explains the motivation behind the use of question generation mechanism, and how it is implemented. It involves several programming techniques, including

A. Recursion
B. Object Orientated Programming, and
C. Selection and Looping

I. Introduction: The Object Approach to Problem Generation

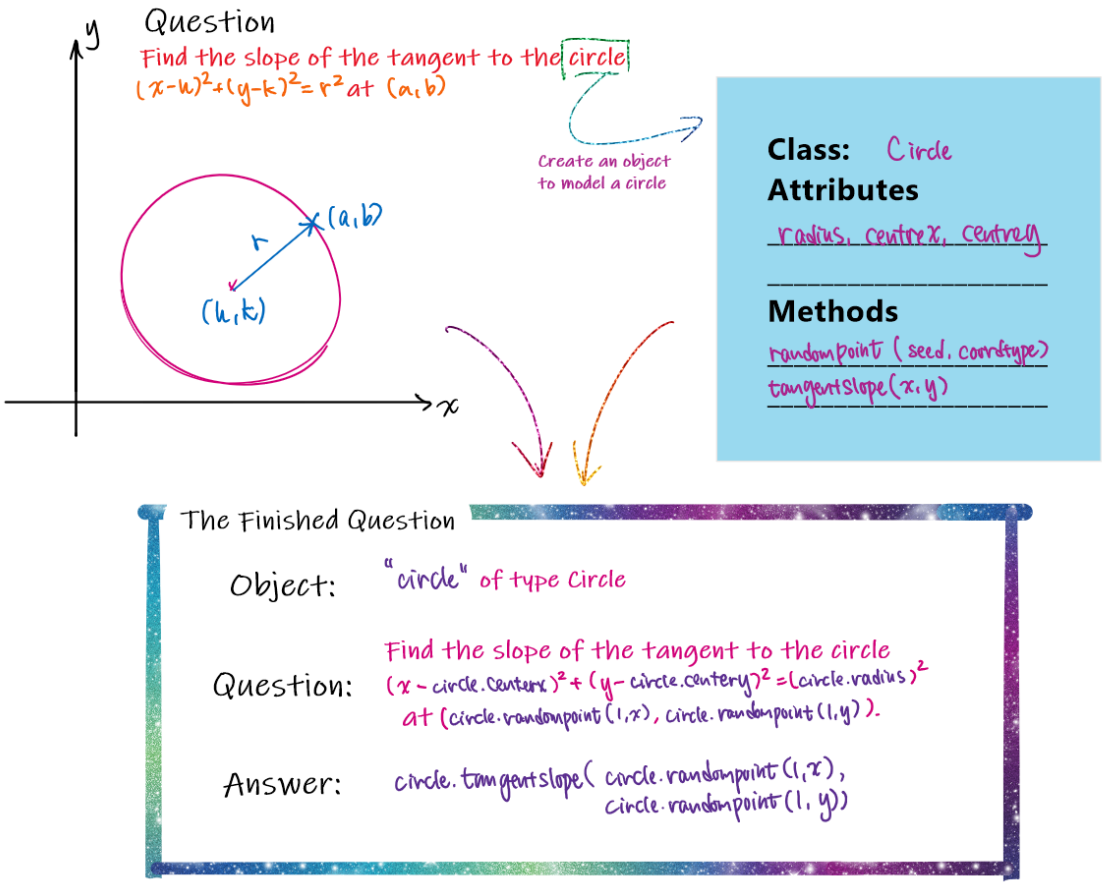
The questions that appear in the game are actually generated in real-time. They are not from long databases of questions with numbers already plugged in. In addition, the questions are stored in a database, not inside the program. When a question is required, a question statement is picked out of a database, and coefficients will be generated and plugged into the statement, generating a whole new question every single time. The implementation of such a system, of course, was filled with ideas and challenges.

In designing the algorithm for this function, I have first identified several characteristics in high school math questions, which will be explained along with questions

Single Constraints	"Evaluate $\frac{e}{f}$." There are two possible single constraints in this problem. - Range constraint: for example, e might be an integer between -100 and 100. - Value constraint: $f \neq 0$, or else it would cause a division-by-zero exception, and this problem would be invalid.
Joint Constraints	"Evaluate $\frac{a}{b-c}$." There are two possible joint constraints in this problem: - Range constraint: for example, $\frac{a}{b-c}$ as a whole must be within a range. Denominators and numerators might also have a range. - Value constraint: $b - c \neq 0$ for the same aforementioned reason. A complicated example of joint constraints is quadratic equations. Owing to the fact that they might have two, one or even no real solutions, it must be well controlled so that real roots do exist. Furthermore, constraints may be added so that the roots must be rational.
Functions	"Evaluate the L.C.M. of a and b." Since there are no simple single-lined formula to evaluate this, a function is needed to evaluate the L.C.M. of a and b.

An initial thought would be to randomly generate coefficients, plugging them into formulas or functions, and asking the generator to regenerate the coefficients if a joint constraint is violated. Several disadvantages would quickly come to mind. We'll talk about them in a bit.

A more advanced idea, which is the one employed in this program, is to use reverse engineering to create the problems. The answers are first generated, and working backwards, a question is set based on these answers. The manifestation of these answers is in the form of mathematical entities, coded as classes, which may be a circle or parabola on a Cartesian plane, or an arithmetic or geometric sequence. Instances if these classes may carry unique attributes, like radius and coordinate of the center for a circle, or first term and common difference for an arithmetic sequence. The question may then be built on top of these attributes to generate a brand new question[15].



▲ A diagram showing the idea of Reverse Engineering with objects.

[15] The inspiration of this idea was from a challenge about generating a Sudoku board. The algorithm used in the solution first generated a complete board, and then slowly took away numbers and checked whether more than one solution exists. Link: *https://www.101computing.net/sudoku-generator-algorithm/*

Object-Orientated Reverse engineering is by far the most efficient method I can think of. Even so, its advantages and disadvantages should be discussed so as to aim for an even better algorithm:

Advantages

When compared with generating random coefficients and validating the question,

- This method is more runtime-efficient. Taking care of joint constraints using the random coefficient method would require many regenerations if the constraints are strict, which will highly reduce the efficiency of the program.
- Moreover, this method is more coding-efficient. The need for functions would require coding one or more function for each question if they all require special treatment. Also, the coding for a single object can enable the creation of many questions.
- To add, this method is more compact and flexible, as long formulae don't have to be typed out for quadratic equations, and questions with varying number of answers can be created. The random coefficient method may require several questions catering for each type of coefficients or answers.

In general,

- This method is more scalable, as the addition of a new object can open the doors to a new type of question.
- This method is more structured, since questions of the same type use the same objects. Debugging would also be easier because changing the implementation of the object will fix all errors within questions which use the object.

Disadvantages

- There are problems which doesn't have a structured Mathematical entity that can be easily modeled into an object, like problems in number theory or calculus, the latter being in the original plan but given up because of this reason. The former would be more easily solved using a formula or a function, suggesting a combination both methods.
- This method does not work well if two objects need to interact, which is a joint-constraint. For example, though a formula can be written to solve the position of the intersection of two straight lines (or a line and a circle), the range of the coordinates cannot be controlled well. Another example is the generation of 2 polynomials such that they have at least factor in common.

  (The latter can be solved by adding setting the roots manually, but a list of randomly generated numbers is needed for this occasion.)
- Requires a lot of brain power to get it to work.

II. Question Format

After settling the concept of the algorithm, we should consider how the algorithm is implemented. In particular, how the objects are specified and how the attributes are accessed. This part is aimed at explaining the format in which question stored, and how these question ask the program to generate and use objects.

Several objects have been defined to create a limited number of questions. There are two types of object types: Mathematical Objects and Processing Objects. Mathematical Objects model entities that appear in Mathematics questions as described in the last part, while Processing Objects are for formatting the question and validating values. We give a full list here:

Mathematical Objects

- `as` (*Arithmetic Sequence*)
- `multias` (*Integer-multiple Arithmetic Sequence*), subclass of `as`
- `gs` (*Geometric Sequence*)
- `polynomial` (*Polynomial*)
- `quadratic` (*Quadratic*), subclass of `polynomial`

Processing Objects

- `randint` (*Random Integer)*
- `fraction` (*Random Fraction*)
- `randfloat` (*Random Floating-Point Number*)
- `array` (*Array*)
- `eval` (*Processing Object*)

For information about the defining parameters, attributes and methods of these objects, please refer to *Appendix A. List of Mathematical Objects in mQuestionObjects.py*.

The questions were first written into a MySQL database so that the organization and editing of the questions are easier. An export of the database is stored in Questions Database/questions_questions.sql. The questions were then converted into a JSON string and stored into files/Questions.json. Each question has the following fields:

Field	**Example** (Taking Sequence: AS Q4 as an example)
`comment` (Description)	`Sequence: AS Q4`
`diff` (Difficulty Level)	`2`
`diffPt` (Difficulty Variation)	`-0.1`
`time` (Time Limit)	`90`
`obj` (Objects)	`{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-300, 300, -30, 30)¥"}, ¥"int¥": {¥"objname¥": ¥"randint¥", ¥"args¥": ¥"(7, 120)¥"}}`
`questionStm` (Unprocessed question statement)	`Find the {int,ord} term of an arithmetic sequence whose first term is {as,first} and common difference is {as,diff}.`
`answerStm` (Unprocessed answer statement)	`{as,valueByNo,{int,value}}`

In the following sections, we will explain each field of the questions.

Difficulty Level and Difficulty Variation

There are 6 difficulties levels for the questions. The first three are easier questions, with each difficulty level corresponding to a certain topic. The latter three are harder questions, and their topics correspond to the first three respectively. The topics of each difficulty level are listed in *1C Area of Learning*.

In the game, the first three will be more often picked during Grinding, while the latter three will appear more often in battles with Mathematicians.

Difficulty Variation describes how much does the actual difficulty of the question differ from the "average" difficulty of questions in the same difficulty level. This difference will be reflected in the extra or reduced damage that the player and the opponents deal.

Objects

`obj` is a JSON string. The name of the objects is set as they keys, and the corresponding values store instructions on on what and how the object should be generated. The JSON string must be generated using the `ObjectJsonGenerator`, which can be accessed by running `mQuestionRead` as a main program.

Here, we discuss the format of each key-value pair. The following example has removed the backslashes (¥) that is required for the program to work. Each key-value pair has three components:

Object Identifier → `"mySequence" :{`
Object Type → `"objname": "as",`
Arguments → `"args": "(-300, 300, -30, 30)" }`

Object Identifier	:	The object identifier can only contain digits and letters.
Object Types	:	A list of object types has already been given in the table above. (Note that it is case-insensitive.)
Arguments	:	Arguments specify the properties of the objects, like numerical range, generation mode and so on. Arguments must be written with round brackets and separated with commas. If the argument is of string type, quotes are required. It is also possible to specify key arguments: `(-300, 300, d2 = 30, d1 = -30)` A detailed list of arguments for each of the object types is given in *Appendix A. List of Mathematical Objects in mQuestionObjects.py*.

Retrieval of Attributes and Methods in Question and Answer Statements

The way that objects are accessed in the statements is through placeholders. The placeholders are comma-separated arguments placed between curly brackets, and the arguments specify **which object** and **what from the object** needs to be retrieved. The placeholders are then replaced with the retrieved information[16].

The following is an example of how the placeholders are written, and how it is converted into a question.

Question: Polynomial: Q4

Object Definition:

```
{"p": {"objname": "POLYNOMIAL", "args": "(3)"},
 "eval": {"objname": "eval", "args": "()"}}
```

	Statement	Result
Question	`It is given that {p,denon,0,_hideone}x {eval,evalInt,-({p,numer,0}),_showsignspace} is a root of the cubic polynomial P(x) = {eval,varDisplay,{p,displayCoeff,3},3,False,True} {eval,varDisplay,{p,displayCoeff,2},2,True,True} {eval,varDisplay,{p,displayCoeff,1},1,True,True} + p. Solve P(x) = 0.`	It is given that x + 4 is a root of the cubic polynomial P(x) = x^3 - 4x^2 - 17x + p. Solve P(x) = 0.
Answer	`{p,ansRoots}`	['-4', '3', '5']

A. Attribute Placeholder

The placeholder format for retrieving attributes is strictly a two-argument list:

`{objName,attrName}`

For example, `{p,ansRoots}` retrieves the roots of the polynomial `p`.

B. Method Placeholder

The placeholder format for accessing methods is an argument list with 3 or more arguments:

`{objName,methodName,arg1,arg2,…,argn}`

Starting from the third argument, they will all be passed into the method as arguments for the method. Another thing to note is that there is no need to worry about data type. They will all be passed into the method as strings, but the methods are coded so that they will turn the strings into an appropriate data type to manipulate with.

For example, `{p,denon,0}` retrieves the denominator of the first root of the polynomial `p`[17].

[16] Do not add unnecessary spaces before commas, since they may cause errors. As a result, curly brackets are reserved characters and should not be used except for writing placeholders

[17] It is so defined because there is no clear way to access a list-type attribute.

C. Display Options

The motivation for display options is to increase the readability and aesthetic of the Mathematical questions. For example, the consider the segment `{integer,value}x = 3`. If the `value` of `integer` is 1, the generated segment would become "1x = 3". The 1 unpleasantly stands out. A way to improve this is to add flags to hide it, as in `{integer,value,_hideone}x = 3`, and hence the better-looking result "x = 3".

Displays options do not qualify as arguments, so adding display options do not increase the number of arguments in the list. There are only three display options available:

`_hideone:`	It will turn "`1`" to "", and "`-1`" to "-".
`_showsign:`	It will display the sign of the number, i.e. "`3`" would become "`+3`".
`_showsignspace:`	It will display the sign of the number and add a space between them.

D. Nested Placeholders

Placeholders can (and only) be placed inside method placeholders such that the outer placeholders can use or process the evaluated result returned by inner ones.

Let's dive into a discussion about placeholder evaluation first: the evaluation of placeholders is done from **left to right, inner to outer**. What each layer of evaluation does is that it replaces the placeholder with a string that carries the evaluated value. For example, "`{integer,value}x`" might return "3x", and "`({fraction,value})x`" might return "(4/7)x"[18].

Therefore, there is no need to worry about the data type of nested placeholders. Let's see an example: the imaginary[19] segment

"`{formatter,formatFraction,{sequence,valueOfTerm,{randomInteger,value}}}`"

will first evaluate `{randomInteger,value}` to return "`3`". The replacement of the placeholder results in

"`{formatter,formatFraction,{sequence,valueOfTerm,3}}`"

and will then evaluate `{sequence,valueOfTerm,3}` to return "`0.333333333`" which becomes

"`{formatter,formatFraction,0.333333333}`"

which will at last become "1/3".

E. Answer Statement

The answer, after being processed, will be evaluated. Therefore, if the expression in the answer is "`{integer1,value}+{integer2,value}`", the result after processing may be "3+4", and the program will evaluate it to become "7".

Lists can also be made with this method. For example, the answer statement
"`"["+{integer1,value}+","+{integer2,value}+"]"`" may evaluate to "[3,4]".

---

[18] However, it should be noted that, the expression "`3{integer,value}x`" might become something like "314x".

[19] There are no such objects with these methods. The names are named to be self-explanatory.

III. Question Generation Algorithm: Recursion

This section is devoted to go through how a question is generated from its placeholder form to a human-readable statement. The flowchart for this algorithm is located here: *Figure 6: mQuestions.QuestionGenerator* in *1G Flowcharts*.

Questions are stored in the JSON file files/Questions.json. The function `mQuestionRead.ReadQuestion` is responsible for reading the question out of the file. It is then stored in the global variable `QuestionDatabase`, where they are sorted by difficulty and organized into a dictionary. Note that they keys are of integer type.

When a question is needed, `mQuestions.GenerateQuestion` is called, and the whole process starts.

A. Question Fetching

`mQuestions.QuestionFetch` is called to select a question out of the `QuestionDatabase`. The `obj`, `questionStm` and `answerStm` are then passed into `mQuestions.QuestionFiller`.

B. List Randomization

`mQuestions.Shuffle` is first called to shuffle a random number list called `mQuestions.randList`[20]. The random number list is used to deal with joint constraints in some questions, as described in Disadvantages in *Introduction: The Object Approach to Problem Generation*.

C. Object Generation

`obj` is passed into `mQuestions.GenerateObjects`, and `objDict`, a dictionary with all of the objects paired to their identifiers, are returned. In `mQuestions.GenerateObjects`, the `objName` of each object is sent into a dictionary `mQuestionObjects.switch`, where the `objName` are paired with the actual class names of the objects.

D. Placeholder Evaluation

The question and answer statement, along with `objDict`, is passed into `mQuestions.Process` where the placeholders in the statement will be evaluated. *Recursion is used in this process*.

During the process, `mQuestions.AttrRetriever` will be called to evaluate a placeholder with no nested placeholders, and `mQuestions.DisplayOptions` will be called to process display option flags. Also, `AttrRetriever` will call `mQuestionObjects.ExecuteFunction` to evaluate methods.

`Process` will raise errors if the generated coefficients do not pass validation. This error is captured by `QuestionFiller`, which will completely scrap the generated objects and start another round of generation. After failing for 1000 times, `mQuestion.QuestionFiller` will raise an error, asking `mQuestion.GenerateQuestion` to fetch another question for it.

E. Post Processing

After the question and answer is generated, post processing will carry out beautifying and data type checks on the answers[21].

---

[20] There is also one called `randIntList`, but it is not used.

Suppose Q(x) is a quadratic function. If Q(-2) = -19, Q(-1) = -21 and Q(-5) = -1, solve Q(x) = -3x.
['3', '-7']

If x = a is the symmetry axis of the quadratic graph $y = x^2 + 13x/2 + 3$, find a.
-13/4

Let S(n) denote the sum of the first n terms in an arithmetic sequence. If S(8) = -4456, S(23) = -21436 and S(n) = -27864, find n.
27

Find the sum to inifinity of the following geometric sequence: -3, -1, -1/3, -1/9, ...
-9/2

If the 2nd and 4th term of a geometric sequence are 3 and 1/3 respectively, find the 5th term of the sequence, given that the common difference is positive.
1/9

▲ Some question generated by `QuestionGenerator`.

Programming Technique: Parsing by Recursion

`mQuestions.Process` uses recursion to evaluate nested placeholders. A brief description of the algorithm will be given here.

`Process` reads the statement one character by one from left to right, and when it meets an opening curly bracket it will enter an "environment". Outside the environment, the read characters are copied to a `returnString`, while in the environment the characters are temporarily stored in `tempString`. When it meets a closing bracket, it exits the environment and `tempString` is sent to `Process` again for another round of reading.

When there are curly brackets remaining in the received string, it means that there are nested placeholders that needs to be evaluating, and the process above repeats. Otherwise, there are no nested placeholders, and the expression will be sent to `AttrRetriever` to evaluate the placeholder. The evaluated result will then replace the placeholder.

A slightly more detailed but still simplified flowchart of `Process` is given here: *Figure 7: mQuestions.Process* in *1G Flowcharts*.

The reason for using recursion is that, the same type of thing (Placeholder) can be thrown into the same algorithm (Placeholder Parsing).

Iteration and Selection: As seen in the flowchart, selection is accompanied by iteration to read the characters in `String` and decide what to do with each one of them.

[21] For some reason, there were `Fraction` objects in the answers which, although would work normally in offline game modes, it would create errors when sent through the internet in the Online Multiplayer Mode. This is why `PostProcessing` converts answers into correct data types.

2. Other Programming Techniques

Other programming techniques used include:

- **Event Programming:** the program constantly tests for events, like mouse clicks and keyboard presses, and executes the corresponding functions.

- **Threading and Multiprocessing:** Threads and Processes were used to run two or more functions in parallel. For example, the program uses threads to play background music, and uses processes to open a server in the background.

Sorting and Searching were not needed for this program, but will be needed for functions like creating user accounts and logging in, or making leaderboards.

Another programming technique, or perhaps a programming practice, is the act of defining constants to store the dimensions of an object on the screen. Since the resolution of the game can be changed, it was necessary to calculate every dimension of an on-screen object based on the screen's resolution. Variables were hence defined to store the results after calculation so that the calculation does not have to be repeated every time the object is displayed.

```
self.topCoord = int(self.TopCoord * self.height)
self.titleFontSize = int(self.TitleFont * self.height)
self.headerFontSize = int(self.HeaderFont * self.height)
self.textFontSize = int(self.TextFont * self.height)
self.explanationFontSize = int(self.ExplanationFont * self.height)
self.menuTextSize = int(self.MenuTextSize * self.height)
self.subHeaderFontSize = int(self.HeaderFont * self.height)

self.titleFont = pygame.font.Font(GlobalVar.DefaultMenuTitleFontString, self.titleFontSize)
self.headerFont = pygame.font.Font(GlobalVar.DefaultMenuHeaderFontString, self.headerFontSize)
self.textFont = pygame.font.Font(GlobalVar.DefaultMenuTextFontString, self.textFontSize)
self.explanationFont = pygame.font.Font(GlobalVar.DefaultMenuExplanationFontString, self.explanationFontSize)
self.menuFont = pygame.font.Font(GlobalVar.DefaultMenuMenuFontString, self.menuTextSize)
self.subHeaderFont = pygame.font.Font(GlobalVar.DefaultMenuHeaderFontString, self.subHeaderFontSize)
self.inputFontString = GlobalVar.DefaultInputFontString

self.titleSpacing = int(self.TitleSpacing * self.height)
self.headerSpacing = int(self.HeaderSpacing * self.height)
self.textSpacing = int(self.TextSpacing * self.height)
self.subHeaderSpacing = int(self.SubHeaderSpacing * self.height)
self.explanationSpacing = int(self.ExplanationSpacing * self.height)
self.textHorzShift = int(self.TextHorzShift * self.width)

self.menuSpacing = int(self.MenuSpacing * self.width)
self.scrollerSize = int(self.ScrollerSize * self.height)

self.menuHeight = int(self.MenuHeight * self.height)
self.menuWidth = int(self.MenuWidth * self.width)
self.menuTextShift = int(self.MenuTextShift * self.width)
```

▲ Display constants in `pyMenu.FrameMenu`[22].

[22] Object attributes were defined to store these result, which might a bit too memory consuming. This is an improvement.

C. Technical Documentation

1. *Program Libraries*

Several external modules were imported to provide basic I/O and data processing functions. The following section will explain what these modules are used for.

i. `pygame`

This library provides capabilities to simplify game-making, including display and keyboard/mouse input. The library uses event programming and an event loop is used to continuously test for events. This library is frequently used by Python programmers to make games, so there are a lot of resources online.

The downside of these modules is that, event loops can be quite complicated to set up, and it also creates a lot of boilerplate code. In contrast, `arcade` is another game programming module which is less popular but provides more concise syntax, cutting down development time. Unfortunately, it was not adapted because the program was half-finished when it was discovered.

ii. `Wave`, `Simpleaudio` *and* `Pyaudio`

`pygame` also provides audio support, but since the executable generated was unable to support its audio system, an alternative is required. `pyaudio` is a sound module that supports streaming, playing while reading the file. `simpleaudio` is required for `pyaudio` to work, and it seems like `simpleaudio` will provide a way to play the data read. wave is needed for `simpleaudio` to work.

iii. `threading` *and* `multiprocessing`

These modules provide the functionality to run two or more functions at the same time. An object, "Thread" or "Process", is created, which holds the target function. It can then be initiated with the `start()` method. The difference between the two is that threads can access global variables, while processes are completely separated from the main program. For this reason, processes have the `exit()` method, while threads can't be abruptly stopped.

iv. `sockets`

The library rovides low-level networking sockets. The sockets have a server and client side, so players must connect to a running server instead of connecting directly. Placing the server in a separate process (multiprocessing) can simulate direct connection.

v. `json`

JSON is "a lightweight data-interchange format"[23]. Data in the program can be easily converted into text and stored in a text file, then later read to convert them back into program variables. Most files in this program stores data in this format.

[23] *https://www.json.org/json-en.html*

vi. *Others*

- **`os, sys`**: used in the main program to temporarily mute outputs in the console.
- **`traceback`**: used to store the stack of traceback into a text file when an error occurs.
- **`random`**: provides randomization functions to generate random values.
- **`math`**: provides math functions like trigonometry, floor and ceiling functions, and constants like pi and e.
- **`fraction`**: provides the `Fraction` class, whose objects have a value that can be converted back into a fraction represented by a numerator and a denominator.
- **`decimal`**: provides the `Decimal` class, which is used for precise calculations.
- **`re`**: provides functions for using Regular Expressions (Regex).

2. *Self-defined Modules*

There are a lot of self-defined modules in the program, stored systematically in separate files. The following is a list of these files, followed by an explanation of what these modules do.

A List of Self-Defined Modules [24]

0A. Main Program
`The Becoming of a Mathematician.py`

0B. Global Modules
`GlobalVar.py`
`mSprite.py`

0C. Main Program Modules
`Main_AdventureMode.py`
`Main_EntranceSequence.py`
`Main_GameSettings.py`
`Main_MultiplayerMode.py`
`Main_PracticeMode.py`

1. Battle Module
`Main_Battle.py`

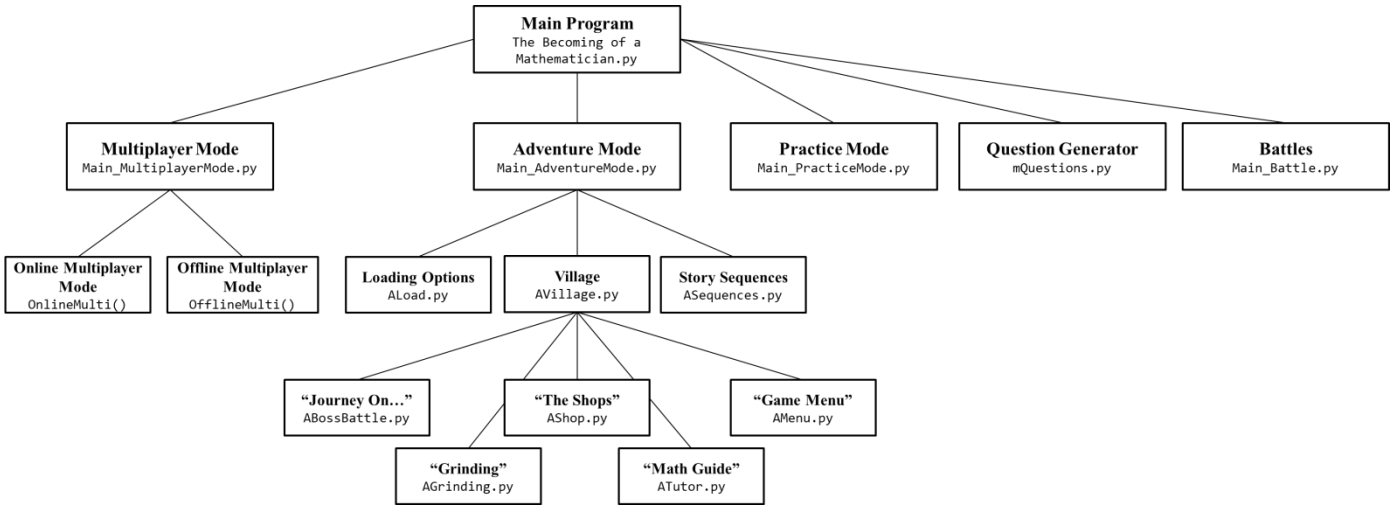
2. Adventure Mode Modules
`ABoss.py`
`ABossBattle.py`
`AGrinding.py`
`ALoad.py`
`AMenu.py`
`APlayer.py`
`ASequence.py`
`AShop.py`
`ATextBox.py`
`ATutor.py`
`AVillage.py`

3. Multiplayer Mode Modules
`mMultiplayerBattle.py`
`mMultiplayerClient.py`
`mMultiplayerObjects.py`
`mMultiplayerPackets.py`
`mMultiplayerServer.py`

4. Question Generation Modules
`mQuestionErrors.py`
`mQuestionFunctions.py`
`mQuestionObjects.py`
`mQuestionRead.py`
`mQuestions.py`

5. Technical Modules
`pyAnimation.py`
`pyFloating.py`
`pyImage.py`
`pyInput.py`
`pyMenu.py`
`pyMouse.py`
`pyText.py`
`pyTextBox.py`
`pyTimer.py`

6. Debugging Modules
`mDebug.py`
`mErrors.py`



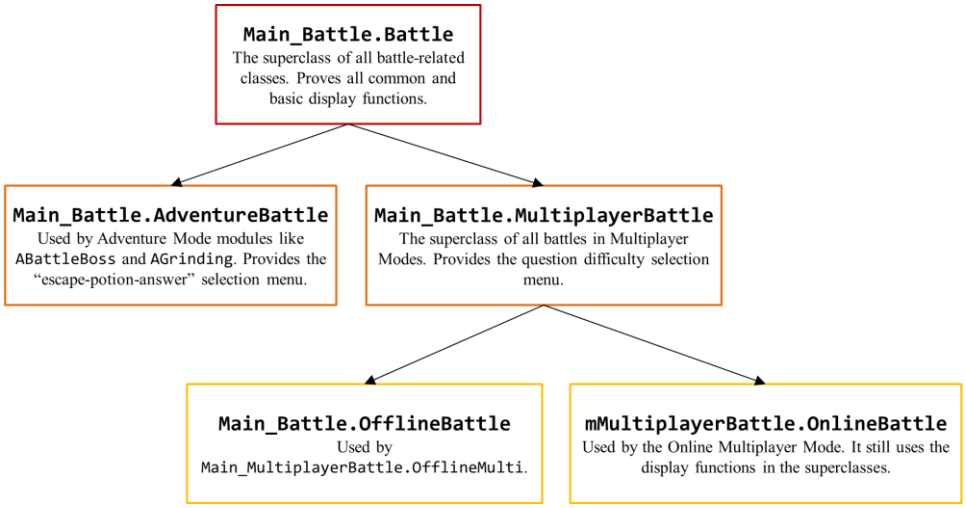
[24] Note:

- 0A, 0C will not be explicitly explained.
- A short note on `mSprite.py`: it contains the `Sprite` class. A Sprite is an entity in the Battle, and two Sprites will fight in a Battle. Two Sprite objects, a player Sprite and the opponent' Sprite, are present in the Battle object, and the Sprites keep track of their states. Sprites have 4 attributes: `name`, `fullHP`, `graphics` and `nowHP`. `nowHP` stores the current HP of the Sprite.
- The contents of `GlobalVar.py` is explained in *2C 3. Organization of Global Variables*.
- `mDebugThread.py` is not used, but not removed for safety.
- `mQuestionWorkplace.py` is not used in the program, but was used in the Debug phase.

1. *Battle Module*

Battles can be regarded as the second-most important game mechanism, because they are responsible for the real "fun" aspect of this game[25]. `Main_Battle` and `mMultiplayerBattle` are two modules which are mainly responsible for the battle mechanism. However, `mMultiplayerBattle.OnlineBattle` in will be discussed in the section *Multiplayer Mode Modules*.

The following is a hierarchy diagram of the battle classes:



Battles are defined as classes. There are a few reasons why I chose to use classes over functions:

- **Variables**: In the `Battle` class, there are instance variables that are called display constants[26]. These display constants need to be shared with functions. Since it would be a hassle to pass in these constants through the arguments of the function, a class can be defined so that its methods can access these variables directly[27].
- **Structure**: It is cleaner to put everything related to battles in a class, instead of having a list of functions lying around the code without explicit structure.
- **Inheritance**: Classes may be inherited so that the subclasses can become more specific in function without the need to redefine the methods one by one, and method can be overridden so that the other methods defined in the superclass can use this new method.

Battles are a sequence of events, like answering a question, attacking and receiving damage, and each of these events are defined as a class method. A `main` method is then defined to use these methods one by one to create a sequence of events. The following documentation will explain the use of some

[25] Of course, battles won't work without questions, so the Question Generatior is regarded as the most important one.

[26] The game has been coded such that resolution can change. Therefore, every dimension of displaying objects, like text, textboxes, images, are all expressed as ratios. These ratios are called display constants, and are class variables. When put into use, instance display constants will be defined though multiplying these ratios to the screen's dimensions. These constants are defined so that calculation is only done once, during the initiation of the object.

[27] It may be argued that using global variables would solve this problem, but it is less structured and difficult to maintain due to weird behavior of global variables.

of these methods[28]. Only the `Battle` and `AdventureBattle` classes will be discussed here. The combined use of `Battle` and `AdventureBattle` in "Journey On…" is shown in this flowchart: *Figure 4: AdventureBattle.Main* in *1G Flowcharts*.

`class Battle`

The highest class in the hierarchy, Battle, provides functions needed to display graphics and answer questions.

`RefreshBattle`	Displays the upper-half of the screen, where the health bar and boss images are located.
`DisplayTextBox`	Refreshes the text box in the lower half of the screen.
`DisplayFloater`	Displays a "floater" (explained later).
`DisplayQuestion() → result`	Displays the question and gets the player's answer.
`DealDamage`	Deals damage to the boss or player.
`Dialogue(speaker, text)`	Displays text that represents talking.
`PromptEscape() → exitFlag`	Asks the player whether to escape.
`LoopBundle`	Ran at all times, and is put inside every loop. It regulates frame rate and more.

[28] Some require more information about how `pygame` displays stuff, which will not be discussed in depth in this report.

`class AdventureBattle`

`AdventureBattle` provides all functions needed exclusively by battles in Adventure Mode.

`Main`	The main function. Each execution corresponds to one answer-attack cycle.
`StartDialogue, EndDialogue`	Displays dialogues at the start and end of the fight.
`Reinitialize`	Replace the existing `bossinfo` with a new one, and reset necessary variables.
`PreSelection, PromptPotionChoose`	Allows the player to use a potion or escape.
`TimeDilation(questionObj)`	Changes the time limit in the `questionObj` based on the player's Time Dilation skill level.
`CalculatePlayerAP(resultDict, timeLimit, diffPt, strengthFlag), CalculateBossAP(diffPt, defenseFlag)`	Calculates damage dealt by the player or the boss.
`FightContinues()`	When neither the player or the boss dies, the fight will continue.
`BossAppears(bossType = "Civilian")`	The boss' entrance scene.
`PromptContinue()`	Asks the player whether he would like to continue grinding.
`Rewards`	Rewards the player for defeating the boss.

2. Adventure Mode (Starts with `A_`)

Adventure Mode is composed of many modules, and they can be divided into two types:
1. **Display Modules**: modules that are directly responsible for display and game flow, and
2. **Background Modules**: modules that work backstage, responsible for things like calculation and data processing.

1. Display Modules

`ALoad`

- **`LoadOptions(screen) → playerInfo, tutorialSequenceFlag, continueFlag`**

Displays the loading options: `tutorialSequenceFlag` plays the tutorial if true.
`continueFlag` returns to the Main Menu if false.

`ASequence.py`

- **`AdventureModeSequence(screen)`** Displays the entrance sequence.
- **`Epilogue(screen) → exitFlag`** Displays the epilogue.

`AVillage.py`

- **`a_Village(screen, playerInfo, sequence = True)`**

Displays the Village. `sequence` specifies whether the tutorial sequence is played. The following modules are used by `AVillage` directly[29]:

`ABossBattle`

- **`a_DoBoss(screen, playerInfo) → exitFlag`** Responsible for "`Journey On…`".

`AGrinding`

- **`a_DoGrinding(screen, playerInfo)`** Responsible for "Grinding". Used by

`ATutor`

- **`a_tutor(playerInfo)`** Responsible for "Math Guide".

`AShop`

- **`a_DoShop(screen, playerInfo)`** `a_DoShop` is responsible for "The Shops". It has several local functions which are responsible for the respective shops.
  - **`class ShopClass`** Its class variables and static functions store product information.
  - **`class ShopMenu`** It is used to display the shop menus.
  - **`class PaymentBox`** It is used to display the text box at the bottom when the player purchases an item.

[29] `exitFlag` specifies whether the player wants to quit the game after beating the final Mathematician.

AMenu

- `a_Menu(screen,playerInfo) → exitFlag` a_Menu is responsible for "Game Menu"
  - `a_QuitMenu(screen) → exitFlag` Responsible for "Quit Game".
  - `a_Save(screen, playerInfo)` Responsible for "Save Game".
  - `a_Stats(screen, playerInfo)` Responsible for "View Statistics".

2. Background Modules

ABoss

`class Boss`

`Boss` stores information about a boss. Their attributes are read out from the data file and assigned to their constructor, and all of them are saved inside a global variable. `BossInfo()` returns a Sprite object to be used by the Battle modules.

Functions

- `WriteBoss(filename)` Displays a CLI interface for creating a new Boss entry. This can be accessed by running `ABoss` as a main program.
- `ReadBoss(filename) → bossDict` Reads out boss information from the specified file.
- `BossFetch(diff, bossDict, index = -1) → bossObj` Randomly selects a boss out of the provided `bossDict`

APlayer.py

Functions

- `Initialize() → (Player) playerInfo` Initializes a `Player` Object with default parameters.
- `LoadSave(filename) → (Player) playerInfo` Initializes a `Player` Object with the data stored in `filename`.
- `SaveFile(filename, (Player) playerInfo)` Writes the player's data into `filename`.

`class Player`

The Player class mainly stores the player's game data. Every field in the files/PlayerData.txt has a corresponding attribute in `Player`, and the class methods are all used to retrieve useful information from the data. A few examples are given below:

- **`Skills(increments = {}) → skills`**

  `increments` is a dictionary where the keys is the names of the skills, and the values are the amount of increment to the skills. `increments` is optional. It will return the player's `skill` dictionary.

- **`Experience(increments) → Dict`**

  Increments the experience level of the player by `increment`, and returns information about the experience requirement of the next level. The experience levels are calculated within the method.

  Keys: **`exp, level, skillPt`**,
  `expLevel`: sum of EXP up to next lvl
  `expReq`: EXP required for next lvl
  `levelUp`: did player level up?

- **`fullAP()`**

  Calculates the full AP from the player's Health skill.

`ATextBox`

`ATextBox` contains classes that provide displaying functions, including text boxes at the bottom of the screen during the entrance and ending sequences and pop-up messages during special occasions, like a game crash. The three classes in the module are called `TextBox`, `Scene` and `PopUp`, responsible for different usage of the text box.

3. Multiplayer Mode Modules

Main_MultiplayerMode

`Main_MultiplayerMode` contains the function `m_MultiplayerMode`. It has many local functions, including the following:

Responsible for Offline Multiplayer: `OfflineMulti`

Responsible for Online Multiplayer: `OnlineMulti`, along with the following functions

- `OpenServer` Opens a server (runs the `mainfunc` function in `mMultiplayerServer`) in a separate process. The `Process` object responsible for this is `serverThread`.
- `Name` Asks the player for his name.
- `ConnectOptions` Responsible for "Connection Options"
- `IPAddress` Asks the player for the IP address of the server to which he wants to connect.
- `ConnectionFailed` If the program fails to join the server, it displays the error message of the problem.
- `Troubleshoot` Lists a bunch of connection error codes and their possible remedies.

mMultiplayerBattle.py

Provides the `OnlineBattle` class, like the ones in the `Main_Battle` module, which has the following methods:

- `Main` The main function.
- `Battling` Responsible for controlling the answer-attack cycle.
- `Result` Display the results (win, loss, disconnected) after the game.
- `Monitor, Update` Ran repeatedly at all times. Monitor gets the game state from the server, and they, together, updates the log box and ends the game when required.
- `GameStart, Penalty, DisplayQuestion` Responsible for displaying parts of the answer-attack cycle.

mMultiplayerClient.py

Provides the `Client` class, which is responsible for all network connections in the Online Multiplayer Mode. All game-related things are done by the `OnlineBattle` class. It has the following methods:

- `connect` Connects to the server.
- `getQuestion` Asks the server for a math question.
- `getStatus` Asks the server for the game object of this game.
- `attack` Requests the server to deal damage to the opponent.

mMultiplayerObjects.py

Provides the `ResultObj` and `Game` objects. The `ResultObj` stores the result after answering a question, and the `Game` object stores the status of the games, including the health of the players, and calculates the damages done based on the `ResultObj` sent by the players.

mMultiplayerPackets.py

Provides the `Send` and `Receive` functions to convert between data and packets. The packets are JSON strings, with the following fields:

`{TYPE: type,` — *`TYPE` is the type of packet. `TYPE` is a string, which can be "`HANDSHAKING`" for the handshaking phase, "`GET`" for getting the Game Object, "`QUESTION`" for getting a question, and "`ATTACK`" for attacking.*

`DATA: sendData}` — *`DATA` is the data associated with the package, which can be empty. In "`GET`", the client sends an "`GET`" package with empty `DATA`, while the server returns a "`GET`" with the game object as `DATA`. The `DATA` is encoded in Base32.*

mMultiplayerServer.py

Contains the `mainfunc` function which is used to open a server. The module also contains a class, `ThreadedClient`, which will be ran in a thread every time a player connects to the server to deal with it (time sharing).

4. *Question Generation Module*

The following section will place focus on the Question Generation functions individually, instead of them working as a whole. For a description of how they work together, please refer to *2B III. Question Generation Algorithm: Recursion*.

mQuestionRead

ReadQuestion(filename) → (dict)	Reads the questions from the file specified by filename, and sorts them by difficulty in a dictionary.
QuestionDatabase	
ObjectJSONGenerator	Generates the JSON string to place in the question's `obj` field.
WriteQuestion(filename)	(Obsolete) Used to manually input questions[30].

mQuestions

QuestionGenerator	Generates a question.
QuestionFetch	Randomly picks a question out of `QuestionDatabase` based on parameter `diff` and returns it to `QuestionGenerator`. Parameter `index` specifies the exact question to be accessed for debugging.
QuestionFiller	Fills in the `questionStm` and `answerStm` with coefficients to generate a question.
AnswerChecker	Checks whether the parameter `userinput` is a correct answer based on parameter `answer`.
GenerateObjects	Generates the objects as specified by the parameter `obj`.
Process	Processes the parameter `string`, which may be the `QuestionStm` or `AnswerStm`.
DisplayOptions	Processes display options of the coefficients.
AttrRetriever	Retrieves the attribute of an object in an `objDict` specified by the parameter `argsString`.
PostProcessing	Modifies the parameter `answer` so that it fits a certain format.

mQuestionObjects

`mQuestionObjects` contains a long list of objects which are used to generate the questions. They will be documented in *Appendix A. List of Mathematical Objects in mQuestionObjects.py*.

ExecuteFunction
Retrieves the result of executing the method `function` of `obj`. Used by Process.

[30] MySQL was then used to store the questions, replacing the need for this function.

5. *Technical Modules (Starts with `py`)*

These modules are difficult to conclude with one phrase. They provide diverse functions that are not related to the content of the game, but instead involve technical stuff like display, timing etc. The following are the main functions of the following modules:

`pyAnimation`	Provides "animation", like the fade in and out effect in menu transitions. Function **`FadeOut`** fades out, **`FadeIn`** fades in, and **`EnterFade`** combines the two.
`pyFloating`	Provides the **`floater`** object. A "floater"[31] is an arrow that oscillates up and down[32], and is used to tell the player to press Enter.
`pyImage`	Provides functions like cropping (**`Crop`**) or scaling an image (**`ScaleGraphics`**). **`Wallpaper`** is used to scale an image for use as a background wallpaper.
`pyInput`	Provides an **`TextInput`** object that allows the player to enter text, which `pygame`, again, does not provide.
`pyMenu`	Provides two objects, the **`SelectionMenu`** and **`FrameMenu`**, to create menus. **`FrameMenu`** uses **`SelectionMenu`** to create simple full-screen menus quicky.
`pyMouse`	Provides the **`InArea`** function, which tests whether the mouse is within an area in the screen.
`pyText`	Provides "text wrapping" (**`TextWrap`**), breaking a long string of text into multiple lines, which is not provided by `pygame`. The list of strings returned by `TextWrap` is then rendered by **`TextRender`**.
`pyTimer`	Provides timer objects, which `pygame` does not provide in version 1.9.6. **`BgTimer`** records time in the background, while **`Timer`** can display the timer.

These functionalities are created by me because `pygame` does not provide these basic game production functionalities, which is one of my biggest regrets for choosing `pygame` instead of `arcade`.

[31] This word is made up. A possibly better alternative is "continue indicator".
[32] Fun fact: the trajectory of the floater follows a sinusoidal path, which uses the math module.

Selected Classes for Documentation

pyInput

class TextInput(font, width, clock, color, initialText, cursorBlinkTime[33], ...)

A `TextInput` object uses keyboard press events to enter characters into a string. The method **`Typing`** will return True when the player has pressed the return key, and **`Text`** will return the text. **`Surface`** will return a `pygame.Surface` object of the input box.

```
inputObj = TextInput(...)              Initiating an TextInput object.

while inputObj.Typing():               While the player has not pressed Enter…
    screen.fill((255, 255, 255))       Refill the screen with white
    screen.blit(inputObj.Surface())    Display the text typed on the screen
    screen.display.update()            Update the display

inputText = inputObj.Text()            Assign the inputted text to inputText
```

pyMenu

class SelectionMenu(scroller, background, scrollerShift = 3, textShift = 3, alignment = "left", spacing = 0, scrollerSize = -1)

`Scroller` is the arrows that points to the selected choice, while `background` is a `Surface` that will specify the size of each choice in the menu. The following is an example of how the class is used.

```
# Create Menu
mainMenu = SelectionMenu(MenuArrow, Background, 0, Spacing)    Defining the menu.
mainMenu.CreateItem("Choice 1", Font, (10, 100))  # 0         Creating the choices. Indices start from 0.
mainMenu.CreateItem("Choice 2", Font, (10, 200))  # 1
mainMenu.CreateItem("Choice 3", Font, (10, 300))  # 2

# Main Loop
while True:
for event in pygame.event.get():
        result = mainMenu.Scrolling(event)                     Receiving the player's choice.
        if result == 0:                                        Testing the player's choice, and executing
            # Something Happens                                    the corresponding function.
        elif result == 1:
            # Something Else Happens
        elif result == 2:
            # Wow! Something Happened
    mainMenu.DisplayItem(self.screen)                          Displaying the menu.
    pygame.display.Update()
```

[33] Any parameters that have to do with time are expressed in milliseconds.

6. Debugging Modules

`mDebug`

`mDebug` contains functions that might alternate the displayed Math question, and store the results to a text file for inspection.

`mQuestionErrors`

The program contains user-defined errors, a very useful function provided by Python's object-like Exceptions. The following is a list of these errors and the conditions for them to be raised:

- `Error` Superclass of all Exceptions defined in `mQuestionErrors`.
- `FractionError` Raised when the numerator or denominator of the evaluated fraction is too large.
- `RangeError` Raised when the value of the evaluated number is too large.
- `ArrayError` Raised when an array cannot generate enough unique values.
- `TypeError` Not used.
- `ValidationError` Raised when the evaluated expression violates a validation condition.

3. Organization of Global Variables

Numerous global variables were defined in the program so that maintenance can be easier. The following section summarizes these variables and states their uses.

i. *GlobalVar.py*

A. Images

Images are either stored as `pygame Surfaces` or file paths (Strings).

Variable Name	Type	Usage
`ProfileImage`	Surface	A `pygame` Surface containing the player's profile image.
`Player2Image`	String	Path to the image for Player 2 in Multiplayer Mode.
`DefaultTextBox (= DefaultBattleBox, DefaultShopTextboxWallpaper, DefaultLogBox)`	Surface	A text box that serves to be the background of text. This box is the big rectangle on the bottom of the screen in Battles. The other variables store the exact same thing as `DefaultTextBox`. They are used in Battles, Shops and Multiplayer Mode.
`DefaultVillageWallpaper, DefaultMultiplayerWallpaper, DefaultMultiplayerWallpaperRare, DefaultGrindWallpaper, DefaultMultiOnline, DefaultMultiOffline`	String	File paths to the wall papers seen in the Battles.
`DefaultBattleWallpaper`	Dict	
`DefaultBattleNameTag`	String	Paths to the blue paint smear behind the names tags in Battles.
`HeartStr, HealthBarBackdropStr, HealthBarOverlayStr`	String	These images are used to create the health bar.
`Heart, HealthBarBackdrop, HealthBarOverlay`	Surface	
`DefaultChestplate, DefaultLeggings, DefaultSword, DefaultShirt, DefaultTrunks, DefaultTwig, DefaultPotion1, DefaultPotion2, DefaultPotion3, DefaultPotion4, DefaultSkills, DefaultShopSoldOut`	String	These images are used in the Shop.
`DefaultMenuStatOne DefaultMenuStatTwo DefaultMenuStatThree DefaultMenuStatFour`	String	These images are used in the "View Statistics" menu.
`MenuArrow, DefaultMenuBackground, DefaultMenuSelectBackground`	Surface	These are images used in selection menus. Most menus use `MenuArrow`. The other two are backgrounds for the choices in selection menus.

Variable Name	Type	Usage
DefaultMenuScroller	Surface	= MenuArrow
DefaultShopTextboxFloater	Surface	= FloaterArrow
DefaultShopScroller	Surface	This is the arrow used in the shop
FloaterArrow	Surface	This is the floating, downward pointing triangle that prompts the player to click Enter.
MenuIcon, Floppy, Settings	String	These are the image shown in the selection menu of the main menu, "Save Game" and "Game Settings".

B. Fonts

Variable Name	Type	Usage
GameFont, BlockyFont, MathFont, ExplanationFont, InputFont, MathInputFont, BattleFontString, DefaultShopTextboxTextFontString, DefaultShopTextboxInputFontString, DefaultShopTitleFontString, DefaultShopBalanceFontString, DefaultShopFontString, DefaultInputFontString, DefaultMenuTitleFontString, DefaultMenuHeaderFontString, DefaultMenuTextFontString, DefaultMenuExplanationFontString, DefaultMenuMenuFontString	String	File paths to font files for fonts used in the game. GameFont is the default font.
FONTHEIGHT	Integer	This is the font used in the Health Bar. FONTHEIGHT is used as a reference to scale the font.
BattleFont	Font Object	

C. Files

Variable Name	Type	Usage
QuestionDatabase	Dictionary	Stores the math question retrieved from the file files/Questions.json.
BossDatabase	Dictionary	Stores the boss information retrieved from the file files/Bosses.txt.
PlayerDataFileName	String	File path to the default save file.
SettingDict	Dictionary	Stores the game settings read from the file Game Settings/GameSettings.txt.
ErrorFileName	String	Path to the error file.
DebugFile	String	Path to the debugging file.
GuideFilename	Dictionary	Stores the file path to the Math Guide files and folders.
GuidePath	String	

D. Sound Files

Variable Name	Type	Usage
`ButtonClick`	`simpleaudio WaveObject`	These are sound effects and background music heard in the background.
`BattleMusic1, BattleMusic2, BattleMusic3, MenuMusic, VillageMusic, AmbientMusic, ShopMusic`	String	
`BattleMusic`	List	
`audio`	`audioClass` Object (self-defined)	A thread object for playing music in the background.

E. Constants and Miscellaneous

Variable Name	Type	Usage
`LevelNames`	Dictionary	Stores the category of questions in each level.

`DefendPotionPerc`
`AttackPotionPerc`
`HealthPotionPerc`
	Real	Stores the magnitude of the effects of potions, e.g. percentage of health recovery.

`CentralClock`	Pygame Clock object	A global clock object for keeping track of the game framerate and timers.
`Color`	Color object (self-defined)	Stores the RGB codes of several colours.
`PlayerHP`	Integer	The default health points in Multipalyer Mode.
`AttackDmg`	Dictionary	The amount of damage dealt from answering question of each difficulty.
`DefaultPort`	Integer	Default Port Number
`bool`	Boolean	Debug Module
`GameFPS`	Integer	The game frame rate is 60 frames per second.

ii. Other Modules

Module	Variable Name	Usage
`mQuestions`	`switch`	A dictionary which matches the `objName` stored in each question to the class names of the objects coded in the program.
`mQuestionObjects`	`DISPLAYUNICODE`	A flag indicating whether Unicode exponents, ², can be displayed instead of "^2"
`mQuestionObjects`	`randList`	A list of randomly generated integers and fractions that will be regenerated each time a new question is being written.

4. Files

The following is a list of text files used by the program and their corresponding formats.

Path	Format	Usage
files/Questions.json	JSON	File for storing questions.
files/Bosses.txt	JSON	File for storing boss information.
files/PlayerData.txt	JSON	File for storing player game data.
files/DebugInfo.txt	JSON	Record of answered math questions, including the time used, result, user input, and the question object.
Game Settings/GameSettings.txt	Text	File for storing Game Settings.
files/ErrorFile.txt	Text	File for storing the traceback (error message) when a fatal error occurs.
files/Guide/One.txt files/Guide/Two.txt files/Guide/Three.txt	Text	A list of useful learning resources for each level of math questions.

File Formats

1. files/PlayerData.txt

```
{
    "skills": {                 Skill level for each level
        "strengthPt": 0,        Integer
        "defensePt": 0,         Integer
        "timePt": 0,            Integer
        "agilityPt": 0,         Integer
        "healthPt": 0           Integer
    },
    "equipment": {              Tier of equipment
        "chestplate": 0,        Integer
        "leggings": 0,          Integer
        "sword": 0              Integer
    },
    "potions": {                  Number of potions
        "regenPot": 0,          Integer
        "attackPot": 0,         Integer
        "defendPot": 0          Integer
    },
    "level": 1,             Experience Level (Integer)
    "exp": 0,               Amount of EXP (Integer)
    "skillPt": 0,           Amount of Skill Points (Integer)
    "bal": 0,               Balance (Integer)
    "bosslevel": 1,         Current math level (Integer)
    "beat": false           Whether the player has beaten the game
}                                   at least once (Boolean)
```

2. files/Bosses.txt

```
{ "Bosses": [
   {
       "diff": 1,                          Difficulty (String)
       "name": "Archimedes",               Name of the Boss (String)
       "fullHP": 3300,                     Initial Health Points (Integer)
       "fullAP": 80,                       Attack Strength (Integer)
       "graphics": ...,                    Path to the image (String)
       "rewards": {                        Rewards:
           "exp": 0,                           Experience (Integer)
           "cash": 0                           Cash (Integer)
       },
       "dialogue": [                       Starting Dialogue
               ["Speaker", "Speech"]
               [...], ...
           ],
       "endDialogue": [                    Ending Dialogue
               ["Speaker", "Speech"]
               [...], ...
           ],
   }, { ... }, ...          *This is a dictionary with "Bosses" as the key and
]                             an array of dictionaries as the value.
```

3. files/Questions.json

```
    [
        {
         "comment" : ...,              Description of the question
                                             (String)
         "diff" : 2,                   Difficulty (Integer)
         "diffPt" : 0,                 Difficulty Variation (Integer)
         "time" : 120,                 Answering Time (Integer)
         "obj" : ...,                  Objects (String)
         "questionStm" : ...,          Question Statement
         "answerStm" : ...,            Answer Statement
        },
        {
         ...                    *This is an array of dictionaries.
        }, ...
    ]
```

4. GameSettings/GameSettings.txt

```
# The resolution of the game.
Resolution: (1200, 800)

# Other settings.
Unicode: True
Music: True
Debug: True
EasyMode: False
```

A comment is preceded by a hash character.
An option. "Resolution" is the name of the option, and "(1200, 800)" is the settings.

These settings only have two values: "True" or "False". "True" can also be represented by "1", and all other values represent "False".

3. TESTING

A. Testing Utilities and Methods

1. Debug Mode

In Game Settings/GameSettings.txt, there are two secret configurations, namely `Debug` and `EasyMode`.

When Easy Mode is turned on, the answer to every question is 1 so that I can quickly skip through the game.

▲ **Easy Mode.**

When the Debug Mode is turned on, there are a few changes to the game:

- **Questions**:
  - The name[34] of the question will be displayed along with the question.
  - After a question is answered, the results[35] will be written into the file files/DebugInfo.txt. The info can be used to improve the questions, and debug question generators and the answer checker.
- **Village**: when F12 is pressed, the data in the file files/PlayerInfo.txt will be written into `playerInfo`. This is so that the game doesn't have to be restarted for the file to reload.

2. Scaffolding

Throughout the debugging process, `print` messages were placed throughout the program so that the flow and program output can be understood clearly. This would help clarify if the program was able to proceed to a certain part of the program, or whether the data to be processed is valid, and hence identifying the reasons for an error. (Usage can be seen in `mQuestion.Process`, `mMultiplayerServer.main`)

However, scaffolding may take some time, as it takes a lot of time to place `print` statements and removing them after debugging is over. It would also be difficult to debug cross-module errors using this method. This leads to another useful tool.

[34] The `comment` field in a question dictionary.
[35] The question dictionary, answering time, user's answer and checking result.

3. Memory Profiler

As its name suggests, this Python library provides output messages to track the memory usage when executing each statement in the target function to be serviced. It will print each executed program statement along with the memory usage in the output. This has assisted me in largely reducing memory by removing badly placed image loading commands in the program.

This module also has a side benefit: since it only displays memory usage for executed statements, it can be used to track the program flow too. As a result, I don't have to place scaffolds for each function I would like to inspect, and has reduced the time used to debug the program. (Example of usage can be seen in `mQuestion.AnswerChecker`)

4. Main Functions

In Python, the selection statement `if __name__ == "__main__":` can test whether the module is imported or ran as the main program. Taking advantage of this, main functions were placed in modules to carry out unit tests. For example, the main program in `Main_MultiplayerMode` opens the game in Multiplayer Mode so that I can concentrate on debugging Multiplayer Mode alone. The main program in `mQuestions` can generate question nonstop to see whether there are any errors as a whole.

5. Workplace

`mQuestionWorkplace` is a separate small program that provides a convenient environment to test out questions or object functionalities. It reads question out of a separate file, files/TestQuestions.json, so I can copy a question to the file and focus on debugging that question.

B. Testing and Debugging

The following section has two parts: Test Cases, and Other Debugging Efforts.

1. Test Cases

In this part, test cases were created to test certain functionalities in the game systematically.

Test 1
Target: Adventure Mode
Method:
1. Play though the game once, trying to find out if it works normally.
2. Follow the following test cases, and see if there are logical erors.

A. Journey On... and Grinding

`boss level`	`beat`	Journey On…	Pass	Grinding	Pass
1	False	Archimedes, Euclid	✓	Monkey, Gorilla	✓
2	False	Isaac Newton, Carl Friedrich Gauss	✓	Insane Salaryman, Insane Office Lady	✓
3	False	Joseph-Louis Lagrange, Leonhard Euler Displays Epilogue when the fight is over.	✓	A Smart Boy, A Smart Girl	✓
3	True	Same as the last case, except when the boss is beaten, the player will be asked whether to play the epilogue again.	✓	Previous Mathematicians, Yandere	✓

B. Player Data

Some fields in the player's play data have limits on how high they can be. The objective of the following test is to see whether there are any methods to break the game by bypassing these extremes. The player data will be incremented (or changed) only with functions within the program, like "The Shops" and "Grinding".

Field	Test Case	Expected	Pass
All	Are the initial data correct?	Correct	✓
`bosslevel` `beated`	Starting from `bosslevel` = 1, defeat all three Mathematicians and do "Journey On..." once more.	`bosslevel` becomes 3, `beated` becomes True	✓
`skills` ≤ 200	Raise skills above maximum level in The Shops.	Remains valid.	✓
`equipment` ≤ 10	Raise equipment above maximum tier in The Shops.	Remains valid.	✓
`level` ≤ 101	Reach maximum level, check statistics.	Shows 101/101, 0 to next level.	✓
	Reach 4105641 EXP, which is the cumulative EXP up to level 102 according to the formula, check statistics.	Shows 101/101, 0 to next level.	✓

C. Math Guide

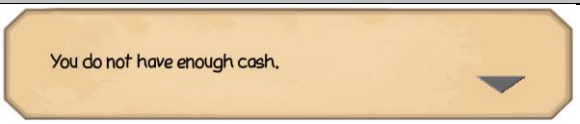
`boss level`	`beat`	File Opened	Pass
1	False	files/Guide/One.txt	✓
2	False	files/Guide/Two.txt	✓
3	False	files/Guide/Three.txt	✓
3	True	files/Guide (Folder)	✓

End of Test 1

The following two tests are case-oriented. The purposes of each case can be divided into 4 types:
(1) Functionality Test
(2) Valid Data (Random)
(3) Extreme Data / Special Case
(4) Illegal Data

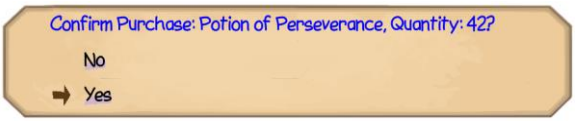
Test 2
Target: Adventure Mode – "The Shops"

A. Armour Shop

Test	Purpose	Expected	Actual	Pass
Try to purchase with no money.	(1)	Displays an insufficient balance message.		✓
Try to purchase with enough money.	(1)	Asks whether you proceed to purchase.		✓
When prompted to purchase, choose Yes.	(1)	Deducts money, and displays a transaction successful message.		✓
When prompted to purchase, choose No.	(1)	Displays a transaction cancelled message.		✓
Purchase the last item in the shop.	(1)	The slot will be replaced with a "sold out" sign.		✓
Try to purchase a sold out item.	(1)	No response.	No response.	✓

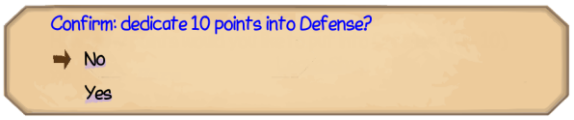
Boundary Conditions				
Test	**Purpose**	**Expected**	**Actual**	**Pass**
Purchase a item with just not enough money ($P - 1$).	(3)	Displays an insufficient balance message.	Displays an insufficient balance message: 	✓
Purchase an item with just enough money (P)	(3)	Asks whether you proceed to purchase, balance drops to 0.	Prompts for purchase confirmation, balance drops to 0: 	✓
Purchase an item with just more than required ($P + 1$)	(3)	Asks whether you proceed to purchase, balance drops to 1	Prompts for purchase confirmation, balance drops to 1: 	✓

B. Potion Shop

Test	Purpose	Expected	Actual	Pass
Try to purchase with no money.	(1)	Prompts for quantity, displays insufficient balance message.		✓
Purchase 1 bottle with enough money.	(1)	Asks whether you proceed to purchase.		✓
Purchase more than 1 bottle without enough money.	(1)	Prompts for quantity, displays insufficient balance message.	Prompts for quantity, displays insufficient balance message.	✓
Purchase more than 1 bottle with enough money.	(1)	Asks whether you proceed to purchase.		✓
When prompted to purchase, choose Yes.	(1)	Deducts money, and displays a transaction successful message.		✓
When prompted to purchase, choose No.	(1)	Displays a transaction cancelled message.		✓

With enough money, when prompt to input quantity, enter				
Test	**Purpose**	**Expected**	**Actual**	**Pass**
0	(1)	Displays a transaction cancelled message.		✓
1	(3)	Prompts to proceed.	Prompts to proceed.	✓
[2, 98]	(2)	Prompts to proceed.	Prompts to proceed.	✓
99	(3)	Prompts to proceed.	Prompts to proceed.	✓
100	(4) Range	No response.	No response.	✓
-1	(4) Range	No response.	No response.	✓
foo	(4) Type	No response.	No response.	✓

C. Skill Shop

Test	Purpose	Expected	Actual	Pass
Try to purchase with no skill points	(1)	Displays insufficient skill points message.		✓
Dedicate 1 point with enough skill points.	(1)	Asks whether you proceed to purchase.		✓
Devote one or more points without enough skill points.	(1)	No response.	No response.	✓
Devote one or more points with enough skill points.	(1)	Asks whether you proceed to purchase.		✓
When prompted to purchase, choose Yes.	(1)	Deducts skill points, and displays a transaction successful message.		✓
When prompted to purchase, choose No.	(1)	Displays a transaction cancelled message.		✓
Try to upgrade a maxed skill.	(1)	Displays skill maxed message.		✓

With enough skill points, prompt to give quantity, enter				
Test	**Purpose**	**Expected**	**Actual**	**Pass**
0	(1)	Displays a transaction cancelled message.	Displays a transaction cancelled message.	✓
1	(3)	Prompts to proceed.	Prompts to proceed.	✓
[2, max – 1]	(2)	Prompts to proceed.	Prompts to proceed.	✓
Maximum points allowed	(3)	Prompts to proceed.	Prompts to proceed.	✓
More than maximum points allowed	(4) Range	No response.	No response.	✓
-1	(4) Range	No response.	No response.	✓
foo	(4) Type	No response.	No response.	✓

End of Test 2

Test 3
Target: Multiplayer Mode – Online Multiplayer Mode

A. Opening a Server

Test	Purpose	Expected	Actual

Enter Nothing	(2)	Open a server with port 40000.	Opens a server with port 40000:
			`Server: Up n' runnin'!`
			`Server: Connect to this server by: 192.168.1.79:40000`

40000	(2)	Open a server with port 40000.	Opens a server with port 40000.

80	(2)	Open a server with port 80[36].	Open a server with port 80.
			`Server: Up n' runnin'!`
			`Server: Connect to this server by: 192.168.1.79:80`

1	(3)	Open a server with port 1.	Open a server with port 1.
			`Server: Up n' runnin'!`
			`Server: Connect to this server by: 192.168.1.79:1`

65535	(3)	Open a server with port 65535.	Open a server with port 65535.
			`Server: Up n' runnin'!`
			`Server: Connect to this server by: 192.168.1.79:65535`

0	(4) Range	Message Appears.	A reminder appears: Accepted Range: 0 < Port Number <= 65535
65536	(4) Range	Message Appears.	A reminder appears.
100000	(4) Range	Message Appears.	A reminder appears.
foo	(4) Type	No response.	No response.

B. Manual Connection

Suppose the Private IP of the computer is 192.168.1.74, and the Physical IP is 119.247.26.21. Here we connect to the server on a computer connected to the same network as the host computer.

Test	Purpose	Expected	Actual

192.168.1.74	(2)	Joins the server opened at port 40000.	Joins the server opened at port 40000.
			
			Client Side: Joined the game and is now waiting for the game to begin
			`Connecting to: 192.168.1.74:40000`
			Online Multiplayer Battle
			Waiting for the game to start...
			Return
			
			Server: Detects a new connection and displays information about the connection
			`Server: =-=-=-=-= New Connection =-=-=-=-=`
			`Server: [IP] ('192.168.1.79', 61889)`
			`Server: [Game] 0`
			`Server: [Side] 1`
			`Server: [Player] 1`
			`Server: [Name] Kwee`
			`Server: =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=`

[36] The actual results are non-deterministic, since ports below 1024 are reserved ports.

Test	Purpose	Expected	Actual
192.168.1.74:40000	(2)	Joins the server opened at port 40000.	Joins the server opened at port 40000.
Open a server at port *X*, and connect by 192.168.1.74:*X*. (*X*∈[1025,65536])	(2)	Joins the server opened at port *X*.	Joins successfully.
Repeat the previous test at port 80.	(3)	Joins the server opened at port 80.	Joins the server opened at port 80.
Repeat the previous test at port 1.	(3)	Joins the server opened at port 1.	Joins the server opened at port 1.
Repeat the previous test at port 65535.	(3)	Joins the server opened at port 65535.	Joins the server opened at port 65535.
192.168.1.74:0	(4) Port Range	An error message appears.	Error code: Invalid Port Number.
192.168.1.74:65536	(4) Port Range	An error message appears.	Invalid Port Number.
foo	(4) Format	An error message appears.	Error code: Invalid IP Format. Format: XX.XX.XX.XX:PortNo. Do not use hostnames.
192.168.1	(4) Format	An error message appears.	Invalid IP Format.
333.12.12.12	(4) IP Range	An error message appears.	Error code: Invalid IP Format. Octets must lie between 0 to 255.
12.333.12.12	(4) IP Range	An error message appears.	Invalid IP Format.
12.12.333.12	(4) IP Range	An error message appears.	Invalid IP Format.
12.12.12.333	(4) IP Range	An error message appears.	Invalid IP Format.
192.168.1.74192.168.1.74	(4) Format	An error message appears.	Invalid IP Format.
192.168.1.74:	(4) Format	An error message appears.	Invalid IP Format.
After Port Forwarding, connect with 119.247.26.21.	(1)	Connects normally.	Connects normally.
Without Port Forwarding, connect with 119.247.26.21.	(1)	An error message appears.	Error code: [WinError 10057] A request to send or receive data was disallowed because the socket is not connected and (when sending on a datagram socket using a sendto call) no address was supplied

End of Test 3

2. Other Debugging Efforts

Practice Mode: mQuestions.QuestionGenerator and mQuestions.AnswerChecker

The main program in `mQuestions` contains code that generates each question 5 times, and checks my input each time. Through this, I have discovered many mistakes in the module and questions, both from the generation and checking of the answers, and hence the accuracy and quality of the questions are improved.

```
437
438     for k in range(6):
439         j = 0
440         run = True
441         while run:
442             try:
443
444                 # For each question, repeat five times
445                 for i in range(5):
446
447                     # Generates a question
448                     question = QuestionGenerator(k+1, questionDict, j)
449
450                     # Print out the question
451                     print(question["comment"] + ": " + question["question"])
452
453                     # Prompt for input
454                     ans = input()
455
456                     # Checks the answer and display the results
457                     print(AnswerChecker(question["answer"], ans), question["answer"])
458
459                     print()
460                 j += 1
461             except:
462                 run = False
463
```

Peer Debugging

I have shared the compiled version of the game with my family and friends so that they can help me play through this enormous program, and help me find subtle but critical errors that would ruin the gaming experience.

I mainly tested out the Online Multiplayer Mode with them, all connecting via my physical IP. The experience has helped me find out critical errors like time out problems, careless data type mistakes and so on.

Buffer

TCP receives messages and stores it in a buffer, not as individual messages. When `recv` (receive) is called on the server's socket, the whole buffer is returned. At first, I have neglected this property of TCP, and called `recv` for each message. I would later find multiple messages combining into one, which would result in an invalid request. The server would then freak out, and the client would also freak out and time out, or raise the "ran out of input" exception. Through testing, I have realized the significance of this property of TCP, and rewritten the server so that when `recv` is called, the returned string is appended to the program's own buffer, and individual requests are isolated from the buffer and processed.

```
54
55              # Receives new messages
56              newStr = conn.recv(4096).decode()
57              Debug("Received string:", newStr)
58
59              if newStr:
60
61                  # Appends it into the program buffer
62                  repos += newStr
63                  Debug("Repository (before):", repos)
64
65                  # Isolate requests from the buffer
66                  processList, repos = Receive(repos)
67                  Debug("Repository (after):", repos)
68                  Debug("Process List:", processList)
```

Data Type

`pickle` is a module that can turn objects into binary string and send it through the network. Through peer testing, I have discovered that some objects, like Fractions, cannot be pickled and would raise an error.

C. Unsolved Bugs

Several bugs have been found in the program, but they are either too difficult to solve, or are insignificant to spend more time to fix.

1. Segmentation Error

At one point, when the program was generating a question in Adventure Mode, the whole game crashed without displaying an error message on the game screen like it should have. The error was the Segmentation Error, which is caused by the program trying to access a restricted memory address, causing the system's file protection to kick into action and shutting down the program. The error has only happened once, two months after the start of the project.

2. pygame.error: Couldn't find glyph

When the program was once put in idle for about a few hours, then when the "pre-selection" menu was displayed in Adventure Mode, the program crashed due to the "Couldn't find glyph" error. The following was the recorded traceback stack:

```
Traceback (most recent call last):
  File "Z:/SCHOOL/ICT/ICT SBA/Program/Workplace/The Becoming of a Mathematician.py", line 141, in <module>
    MainMenu(Screen)
  File "Z:/SCHOOL/ICT/ICT SBA/Program/Workplace/The Becoming of a Mathematician.py", line 97, in MainMenu
    AdventureMode.m_AdventureMode(Screen)
  File "Z:\SCHOOL\ICT\ICT SBA\Program\Workplace\Main_AdventureMode.py", line 19, in m_AdventureMode
    AVillage.a_Village(screen, playerInfo, tutorialSequence)
  File "Z:\SCHOOL\ICT\ICT SBA\Program\Workplace\AVillage.py", line 115, in a_Village
    exitFlag = ABossBattle.a_DoBoss(screen, playerInfo)
  File "Z:\SCHOOL\ICT\ICT SBA\Program\Workplace\ABossBattle.py", line 60, in a_DoBoss
    battleResults = BattleClass.Main()
  File "Z:\SCHOOL\ICT\ICT SBA\Program\Workplace\Main_Battle.py", line 517, in Main
    resultDict = self.DisplayQuestion(questionObj)
  File "Z:\SCHOOL\ICT\ICT SBA\Program\Workplace\Main_Battle.py", line 225, in DisplayQuestion
    pyText.TextWrap(
  File "Z:\SCHOOL\ICT\ICT SBA\Program\Workplace\pyText.py", line 40, in TextWrap
    nowWidth = font.size(j)[0]
pygame.error: Couldn't find glyph
```

3. Freezing in Online Multiplayer Mode modules

When opening a server or connecting to a server, the game would sometimes freeze. Pressing Clrt+C in the console unfreezes the program.

4. Unanticipated Host Error (Audio)

Sometimes, the audio thread would shut down because of the an "Unanticipated Host Error". As a result, there will be no game music.

5. Start-up Menu Controls

In the start-up menu, when prompted to press Enter, it is possible to enter any key to move on to the next message[37].

[37] This is due to insufficient testing. This is an improvement.

6. (Critical) Unsupported data types for JSON in Online Multiplayer Mode

There are a few objects that cannot be packed into a JSON string, and this includes the Fraction object, mentioned before, and also “sets”, which is a special data structure in Python and which is the type that caused the program to fail. Whenever such an object is packed into JSON, the program raises an exception, which is thankfully caught by the “Connection Error” screen and not the game crash message.

Exception handling should be done for a general case, instead of just for the Fraction object. This error was caused by two things: first, a lack of thorough debugging of every question in Online Multiplayer Mode, and second, a lack of data type control in the answers.

4. CONCLUSION

A. Improvements

1. Feedback

The game was distributed to several students for evaluation. I have collected several responses from them, including

- The questions while appropriately difficult for the consolidation of knowledge are rather boring.
  - Improvement: (1) Write more interesting questions that might include application and new, instead of textbook-like consolidation exercises. (2) Enhance the appeal of the game by adding other interesting game elements. (3) Attempt another approach to consolidation.
- (The game is) Excruciating, but overall it's fine.
  - Improvement: (1) It is observed that the questions may be quite tedious instead of light. Questions that are effective in building Mathematical foundation but easy to solve should be written instead. (2) Scenarios like solving difficult problems that are easy to get wrong by careless mistakes, doing the same question again without know how to improve, may cause them to become hopeless and turn away from Math. Therefore, careful guiding and feedback is needed to prevent their rejection in Math.
- The equations are difficult to read.
  - Improvement: (1) Use equation typesetting software, like Latex, to display the equations.

This game was also demonstrated on the project presentation, and I have received feedback from my peers, including,

- Wonderful Interface, a Wide Variety of Game Modes, Detail Instructions, the Multiplayer Mode is a good addition to the game
- Placing a timer in Practice Mode
  - This is against the reason for implementing the Practice Mode: a place where the player can directly access the Question Generator, and not somewhere at which the player is pressured to answer the questions. However, it also might be better since the Practice Mode does seem empty.
- The questions are too difficult, **the players wouldn't want to play the game**.
  - This is true: I can't really imagine students using Wolfram's question generator for practice. The question generator would serve to be more useful for teachers.
- **Unthorough testing**, leading to the "set object not JSON seriable" error showing up during demonstration.
  - Improvement: (1) Test more thoroughly. (2) Keep track and constrict/control the data types of the answers.
- **Where is the "game"**?
- Allow gaming fullscreen. Buttonize the words. Math Guide. Graphics. Add descriptions to explain the use of potions and etc.

In the peer scoring, marks were deducted from Objective (some even gave as low as 3), Testing (mainly because of that data type error), Algorithm and I/O (interface can be more vivid). This is seen by the above suggestions.

2. Reflection

I have discovered several improvements and future development plans in game content and programming.

A. Game Features

A few additions to the game may make it more lively and fun[38]:

1. **Achievements**: by setting achievements, the players may aim to collect them and train math in order to do so.
2. **Hints and Feedback**: if the player gets a problem wrong, or if they do not know how to do the problem, it may make them frustrated and quit the game. Hints should be given so that they wouldn't be stuck forever, and feedback can help them discover what mistake they may have made so that they won't repeat it again.
3. **More variation in the Battles**: presently, the battles are just 1 on 1, and the boss just attacks at a fixed strength you until either of you dies. This may be a bit boring, so to spice up the game, special things may be added. For example,
   - The boss may have special actions, like dodging your attack, poisoning or stunting you, summoning minions (helpers) and so on, such that more strategy is needed to fight the boss.
   - The player may have more skills, or equipment with special effects. For example, besides increasing your attack strength and damage tolerance, it can have thorns with attack back when the boss hits you, it can become invisible so the boss might deal less damage, it may electrocute the boss to stunt him, and so on.
   - In the final fight, special events may happen. For example, after defeating the boss, he suddenly reincarnates into a stronger boss, and the player also has to defeat him. In addition, the boss may throw you a Sudoku puzzle and have you solve it.
4. **Minigames**: Math-related minigames can be added to the game as an alternative way (but not a complete replacement) of earning money.

B. Question Generation Algorithm

There are many improvements that can be made to the algorithm, some of which are discussed below:

Code-like Definition

Every programming language is written based on another language, whether it is Python (based on C), or C (probably machine code). Therefore, it is logically possible to define a set of syntax to "initiate" Mathematical objects, define, change and define their properties, and to declare the question and answer statements. The question file would then only contain its name, time and difficulty, along with the code snippet. A form of design may be as follows:

[38] Sources of inspiration: Bookwork Adventure, Tower of Saviours, Minecraft etc.

```
1  sequence = ArithmeticSequence(arguments)          Declaring an ArithmeticSequence object.
2  term = RandomNumber(5, 10)                        Using the built-in function RandomNumber.
3  question = "What is the "                         Creating the question.
4  question += Ordinal(term)                         Using the compact concatenation/addition operator "+=".
5  question += " of the arithmetic sequence with first term"
           + String(sequence.first)                  Concatenating strings with the "+" operator
           + " and common difference", sequence.diff, "?"
                                                     Concatenating numbers to a string with the "," operator
6  answer = sequence.valueByNo(sequence.ValueOfTerm(term))
                                                     Creating the answer.
7  ? AbsoluteValue(answer) < 300                     Validating this statement with the "?" prefix.
8  answerCheckMode = 2                               It is perfectly expandable to add other attributes to the question.
```

A small program to generate the question, "What is the *n*th term of the arithmetic sequence with first term *a* and common difference *d*?".

There are advantages to this design compared with the present implementation as shown in the above sections:

- It is scalable and expandable: other attributes may be added.
- It is more general: almost everything that can be done in a programming language can be done here, so any algorithm to calculate the answer can be put in here. Objects may not even be used, making number theory questions more easy to make.
- It is more rigorous: type checking and preservation can be done here, while only strings can be used in the present implementation, so it is less prone to errors due to wrong types.

Disadvantages:

- Perhaps an even longer documentation and training.
- Takes a bit more time for coding, but it compensates for the frequent errors used in the present implementation where everything is very specific and will leave errors, making debugging even more time-consuming.
- Takes even more time to generate a question.

Algorithm for Function Generation (for Calculus problems)

Reference: *https://www.marksmath.org/visualization/integration_vs_differentiation/*

The linked website has implemented an algorithm for generating random functions for differentiation and integration. To implement calculus problems, there are two approaches, each with their benefits and limitations:

1. Function Generation: a function is randomly generated for the problems.
   - Advantages: A wide range of problems can be produced.
   - Disadvantages: The exact difficulty of the problem cannot be controlled.
2. Database: a long list of functions is created manually, and the problem just randomizes the coefficients.
   - Advantages: control difficulty and quality of the questions.
   - Disadvantages: tedious, and requires expertise in problem creation to produce this many questions.

Symbolic Expressions as Answers

The definition for equal functions is

> **Equal Functions**
> $f = g$ if $f(x) = g(x)$ for all values of x on the whole domain[39].

To accept symbolic expressions as answers, two modules must be developed:

1. Expression Parsing and Storage Module: A representation method for Mathematical expressions must be developed so that it can be stored as the answer of a question, and so that system can correctly understand the expressions to compare if they are equal. The modules must also know how to use the expression, including substituting values into it.
2. Equality Checking Module: Using the definition above, we check for several values (about 50) of x if the two expressions, the official answer and the answer from the user, are equal.

Question Probabilities

The possibility of choosing a question is equal within a difficulty. This poses a problem: when a problem has many variations (like "integer", "fraction", "positive", "negative" for the common difference in an arithmetic sequence) and is written inside one compact question statement, the variations rarely appears. At the same time, similar and simple questions may be separated into several cases (like asking for the "first term", "common difference", "*n*th term" of a sequence), or differentiation, calculus questions with different forms, will appear very often.

To ensure balanced exercise, it should be made such that these questions appear with different probabilities. For the arithmetic sequence example, the former question should (or may) appear 4 times more often than the latter. As a standard, the "number of variations" or a "relative probability" of the question may be used to describe this attribute.

Geometry Questions

Geometry is another type of Mathematics problem that has lots of variations and is difficult to master, so it is really worth training students to enhance their geometry skillset.

To represent geometry questions, the layout of the game must be rethought since the present text box is not large enough to display the figures at a reasonable size. Also, it can be time consuming to come up with many geometric problems, as for now, there is no way to randomly generate a geometric figure and make a problem on it. Even so, geometry problems would be an attractive addition to the game's question set.

[39] Their domain and codomain sets must also all be the same. Adapted from Wikipedia.

C. Programming Practices

Being an amateur programmer, I have tried my best to adopt the best programming practices so as to make my program maintainable, but due to many factors, I have failed to do so. Here are some of my reflections on my poor programming styles:

1. Comments: **The largest flaw in my program is the lack of comments, an important form of documentation**. There was a large break during the program development, and to pick up programming after a while was challenging as there was absolutely no form of record of what I did before, leading to difficulty in maintenance and debugging.
2. Data types: Even if the program does not enforce type-checking, **it is an important practice to note down what data types are used in the program**. Inconsistency of data types are caused by the lack of type-checking in Python. Integers and characters can be easily mixed up, causing an error in the Online Multiplayer Module that took a whole hour to solve.
3. Thoughtless Programming: When changing just a little re-generation condition in the Question Generator, the other parts immediately created error that implied that the questions were dysfunctional (refer to the Question Generator part), which took a whole day to solve (I looked through every question again, then discovered the problem was not in the questions). **Therefore, think before changing anything in the program, or else it might lead to very bad consequences.**
4. Independency between modules: To further prevent the situation above, modules should be more independent, i.e. more abstraction is needed. When using the modules, I shouldn't have to think of how the modules are implemented, or else it would lead to long coding time. Therefore, I should **put more effort into perfecting the module**: not completely, but to a point where I don't have to worry about it anymore. Also, **write an interface documentation for the module**.
5. Overdependence on Syntax Checking: I tend to make a lot of syntax mistakes when programming, and I rely on the program translator to point out my mistakes. Not only is this bad to my brain, it also wastes a lot of time.

Overall, program development took 3 to 4 full weeks (i.e. 16 hours of work, 8 hour of sleep). This is the consequence of my bad programming techniques, and I would really like to improve.

D. Other Improvements

1. Add a status bar in the village, so that the player doesn't have to go to statistics to check his balance and experience level every time.
2. Allow the player to change the game resolution anywhere in the game.
3. Unload (some) previous menus and loaded images when going to a new one.
4. Allow several save files.
5. Use `arcade` instead of `pygame`.
6. Save previous names or IP address in Online Multiplayer Mode.
7. When the player disconnects in Online Multiplayer Mode, allow a buffer period for the player to log in again and rejoin the game.
8. Shorten program start-up time.
9. Improve program layout and the battle interface, so that more information (text) can be fit onto the screen.
10. Add encoding to the player save file, so that user's cannot just change the numbers to cheat in progress.
11. Link a database directly to the program instead of exporting the contents as a JSON string.
12. Create an interface and a tutorial such that players can create their own math questions using the Question Generator.
13. Create user accounts so that players can have personalized save files.
14. Improvement in game design: it is possible to put all skill points in the same skill to speed through the game. This is due to bad game design and overly powerful skills.
15. Allow the player to deduct skill points, perhaps with a cash penalty.
16. Phrasing and word choices: a lot of game jargons have already been create to describe commonly used game elements. However, I have not searched these up and used those of my own, leading to descriptions that are unclear and unprofessional (it just sounds bad and I don't like it). Therefore, research should be done beforehand.
17. Research: a lack of research has led to such a messy game design. These websites have good tips on how to make a game, and common game elements that come with them: *http://howtomakeanrpg.com/a/how-to-make-an-rpg-textbox.html*, *http://acagamic.com/game-design-course/the-formal-systems-of-games-and-game-design-atoms/*.
18. Boss IDs: the order of appearance of the Mathematicians only depends on its precedence in the boss info file. Boss IDs may be added if a lot of bosses are going to be made, to make the management easier.
19. More story: the story of the game is just a straight line to the finish, without any development. This doesn't not make the game a good RPG. Therefore, a story may be planned, so that there are branches, alternative endings and many more that can involve the player.
20. Batter planning: the behavior of the input module, selection menus using mouse clicks and so on should be planned beforehand so that (1) it is easier for the player to get used to the controls by following the game manual, and so that (2) it is easier to program.
21. Colour coding and formatting within a line of text may be added to emphasize certain words.
22. Text colour throughout the program should be planned so that it is more uniform and organized.
23. Auto saving may be done so that progress will not be loss even if the game crashes when the player has yet to save the game.
24. A method to store the state of a battle, for the same reason in the previous point.
25. Improve the Online Multiplayer Mode status bar so that it is clear when the attack was made.

26. Add game mechanics to the Online Multiplayer Mode. The battles right now are just players answering questions on their own and seeing who's better at math. It purely a battle of wits. To make it more interesting and exciting, elements that allow the interaction between the player, and elements that involve strategies and tatics should be added.
27. "Status effects", like those in Minecraft, may be added.
28. In Debug, append the answer to the question so that it is easier to move through the game without having to use the non-standard "Easy Mode" question. (The program was modified after the official deadline of the program, so that it is easier to be demonstrated to my classmates in the final presentation.)
29. Allow fullscreen in Settings.
30. Do testing more extensively and seriously. (☺)
31. More humor. (😐)

B. Conclusion

1. Before the Presentation

The SBA is, to (probably) a lot of people, a burden. However, to me, it's a valuable chance to learn.

Taking the SBA as an excuse, I used three to four weeks to write the whole program, create the questions and test the game with my friends. In the process, I have tried a lot of new things, like the Python programming language, JSON, Regular Expressions, Concurrency, Event Programming and much more. In making the questions, I have tried using MySQL to store them, and even discovered that the database can directly be hooked up to Python. While trying to make this report look prettier, I have discovered many Microsoft Word functions, like styles, objects, Quick Parts, bookmarks and so on, many of which I have never thought of using.

Making a game has always been a childhood dream, and since I have entered ICT I have been looking forward to doing this. It's a dream come true to see that my game is up and going, and friends having fun while playing with me.

I feel fulfilled: despite the sleepless nights finishing program modules, and being frustrated when the program doesn't work, I have enjoyed the process very much.

2. After the Presentation

It was an interesting experience to show my program to my classmates, and seeing it crash live on screen. This shows that there are still many things I can further improve, to create something that is functional and acceptable to the general audience.

* * *

It actually hurts quite a lot to receive response on how bad the game design and objectives are, even if I understand it in heart. As a math freak (kind of), even I understand myself that the game is boring. My work has always been like so: I always forget what I'm making this game for.

The biggest improvements needed, perhaps, is actually to spend more time on perfecting the game design (so that it complys with the actual user requirements), and to have braveness to scrap the whole project even if its half-finished (I have considered redoing the project multiple times, since I have already spotted this flaw in the game design early on in implementation). If I really was devoted to the project, this would have been quite easy to do.

Things in the past are in the past, and there is no point in crying over spilt milk. The only thing I can do now, is to learn from this past mistake, and remind myself not to repeat it.

Also, if I'm only aiming for a high SBA score, there was never any need to complicate the program so much.

End

APPENDIX

Appendix A. List of Mathematical Objects in `mQuestionObjects.py`

1. *Mathematical Objects*

`as` Arithmetic Sequence

Definition `as(int: a1, int: a2, int: d1, int: d2)`
- `a1, a2`: Range of first term
- `d1, d2`: Range of common difference

Attributes

`first`	:	First Term *a*
`diff`	:	Common Difference *d*
`sign`	:	Sign of *d*
`inequality`	:	"greatest" if $d > 0$, "least" otherwise
`limit`	:	"less than" if $d > 0$, else "greater than" otherwise

Methods

`valueByNo(n)`	:	Value of the *n* th term
`sumByNo(n)`	:	Sum up to the *n* th term
`Product(l, u)`	:	Product from the *l th* term to the *u* th term

`multias` Integer-multiple Arithmetic Sequence
(Subclass of `as`, generates a sequence with terms 0, d, 2d, 3d...)

Definition `multias(int: l, int: u)`
- `l, u`: Range of common difference

Attributes All attributes of `as`.

`base`	:	Common Difference *d*

Methods All methods of `as`.

`gs` Geometric Sequence

Definition `multias(str: type = None, list[int, 2]: spec = None)`
- `type`: "`int`" specifies an integer common ratio *r*, while "`dec`" is a fractional *r*. Not specifying generates both with equal probability.
- `spec`: a list with *(a, r)*. Directly specifies the first term *a* and *r*.

Attributes

`first`	:	First Term *a*
`ratio`	:	Common Ratio *r*
`sign`	:	Sign of *r*

Methods

`valueByNo(n)`	:	Value of the *n* th term
`sumByNo(n)`	:	Sum up to the *n* th term
`sumOdd(n)`	:	Sum up to the first *n* odd-numbered term

`polynomial` *Polynomial*

Definition

```
polynomial(int: deg, frac: leadingCoeff = 0,
list[int, 4]: fractionSpec = (-2, 2, 5, -5),
list[int, 2]: intSpec = (-7, 7),
str: genMode = None, list[frac]: roots = None)
```

- **`deg`**: degree of the polynomial
- **`leadingCoeff`**: leading coefficient of the polynomial
- **`fractionSpec`**: arguments for generating a fractional root (see `fraction`)
- **`intSpec`**: arguments for generating an integer root (see `randint`)
- **`genMode`**: "`int`" specifies an integer common ratio *r*, while "`dec`" is a fractional *r*. Not specifying generates both with equal probability.
- **`roots`**: explicitly specify the roots, pass in a list of length `deg`

Attributes

`roots`	:	Roots (A list)
`deg`	:	Degree of the polynomial
`leadingCoeff`	:	Leading coefficient of the polynomial

Methods[40]

`root(n)`	:	The *n* th root
`denon(n)`[41]	:	The denominator of the *n* th root (1 if the root is an integer)
`numer(n)`	:	The numerator of the *n* th root
`coeff(a)`	:	Smallest integral coefficients
`displayCoeff(a)`	:	Coefficients such that the leading coefficient is that specified at initiation.
`value(x)`	:	Value if `x` was substituted

`quadratic` *Quadratic*

(Subclass of `polynomial`*)*

Definition

```
polynomial(frac: leadingCoeff = 0, str: genMode = None,
list[int, 4]: fractionSpec = (-2, 2, 5, -5),
list[int, 2]: intSpec = (-7, 7),
list[frac]: roots = None)
```

- Refer to *Polynomial*.

Attributes

`rootSum`	:	Sum of Roots
`rootProduct`	:	Product of Roots
`symmetryAxis`	:	The *x*-position of the symmetry axis
`vertex`	:	The *y*-coordinate of the vertex
`discriminant`	:	Discriminant
`limit`	:	The extreme of the quadratic ("maximum" or "minimum")

All attributes of `polynomial`.

Methods All methods of `polynomial`.

---

[40] Both *a* and *n* starts from 0. *a* is the degree of the term (For example, $a_3x^3 + a_2x^2 + a_1x + a_0$).

[41] This is a misspelling that was too late to be found.

2. *Processing Objects*

randint Random Integer

Definition `randint(int: lower, int: upper, int: elower=0, int: eupper=0)`

- **lower**, **upper**: Range of the integer
- **elower**, **eupper**: Range excluded

Attributes

`lower`	:	Lower limit of the integer
`upper`	:	Upper limit of the integer
`value`	:	Value of the integer
`sign`	:	Sign of the integer
`ord`	:	Ordinal representation of the integer (e.g. 1st, 2nd, 3rd, 4th…)

fraction Random Fraction

Definition `fraction(int: lower, int: upper, int: elower=0, int: eupper=0)`

- **valLower**, **valUpper**: Range of value
- **numLower**, **numUpper**: Range of magnitude of numerator and denominator

Attributes

`numerator`	:	Numerator of the fraction
`denominator`	:	Denominator of the fraction
`value`	:	Value of the fraction

randfloat Random Floating-Point Number

Definition `randfloat(int: lower, int: upper, int: precision=3)`

- **lower**, **upper**: Range of the integer
- **precision**: Number of significant figures

Attributes

`lower`	:	Lower limit
`upper`	:	Upper limit
`value`	:	Value
`precision`	:	Number of significant figures

array Array

(Generates an array of mutually different objects of the same type.)

Definition `randfloat(str: objType, int: upper, int: precision=3)`

- **objType**[42]: Type of object stored in the array. Only `"randint"`, `"randfloat"` and `"fraction"` are supported.
- **num**: Number of objects stored in the array.
- **args**: Argument for generating the objects

[42] Add quotes ("") around the object type.

Methods[43]

`attr(n, attr)`	:	Retrieves the attribute *attr* of the *n* th item
`sortedAttr(n, attr)`	:	Retrieves the attribute *attr* of the *n* th item with their values sorted in ascending order
`func(n, func, args)`	:	Executes the method *func* of the *n* th item, passing the *args* into the function.

`eval` *Processing Object*

Definition No arguments are needed.

Methods[44]

- **`eval(string), evalFloat(string)`**
  Returns `eval(string)`.

- **`evalInt(string, max = 10000000)`**
  Evaluates the expression stored in string. The program will restart generation if the magnitude of the integer is greater than `max`.

- **`evalFraction(string, vmax = 100000, dmax = 50000)`**
  Evaluates the expression stored in the string and expresses it as a function in the form of *a*/*b*. Restarts generation if the value is greater than `vmax` or the denominator/numerator is greater than `dmax`.

- **`evalRand(lower, upper, elower = 0, eupper = 0)`**
  Generates a random integer at runtime. See `randint` for the arguments.

- **`evalRandStr(strList)`**
  `strList` is a list of strings. The method randomly chooses one string from the list.

- **`validate(condition)`**
  Restarts generation if the evaluated condition is not satisfied, or if the evaluation raises an exception.

- **`varDisplay(coeff, exponent = '1', sign = False, space = False, varName = "x", vmax = 100000, dmax = 50000)`**
  Returns a string representing a term in a polynomial with degree `exponent` and coefficient `coeff`. The `coeff` is first turn into a fraction, and validated by `vmax` and `dmax`. See `evalFunction` for more information.
  `sign`: adds a sign in front of the coefficient, regardless of positive or negative
  `space`: adds a space between the sign and coefficient
  `varName`: replaces `x` with something else

Examples[45]

Arguments	Result
`(1, 1)`	`x`
`(2, 1)`	`2x`
`(0.6, 2)`	`3x^2/5`
`(1, 0)`	`1`
`(0, 35)`	`(blank)`
`(15, 2, True)`	`+15x^2`
`(-1.5, 2, True)`	`-3x^2/2`
`(15,2,True,True)`	`+ 15x^2`
`(-1.5,2,False,True)`	`- 3x^2/2`
`(3,4,True,True,(3+x/4))`	`+ 3(3+x/4)^4`

[43] *n* starts from 0.
[44] Both *a* and *n* starts from 0. *a* is the degree of the term (For example, $a_3x^3 + a_2x^2 + a_1x + a_0$).
[45] Exponents (^2, ^4) can be replaced with Unicode versions (², ⁴) if Unicode is turned on in Game Settings/GameSettings.txt.

Appendix B. File Directories[46]

- **assets**
  - BattleImage
    - BattleFault.jpg
    - BattleGrind.jpg
    - BattleMultiOffline.jpg
    - BattleMultiOnline.jpg
    - BattleNameTag.png
    - BattleOne.jpg
    - BattleTextBox.png
    - BattleThree.jpg
    - BattleTwo.jpg
  - BossImage
    - Archimedes.png
    - Boy.png
    - CarlFriedrichGauss.png
    - Euclid.png
    - Girl.png
    - Gorilla.png
    - IsaacNewton.png
    - Lagrange.png
    - LeonhardEuler.png
    - Monkey.png
    - SmartBoy.png
    - SmartGirl.png
    - Yandere.png
  - EndingImage
    - Original Files
    - PNGs
    - Scene0.jpg
    - Scene1.jpg
    - Scene2.jpg
    - Scene3.jpg
    - Scene4.jpg
    - Scene5.jpg
    - Scene6.jpg
    - Scene7.jpg
  - Fonts
    - AsapBoldItalic-2jaK.ttf
    - BlockyFont.ttf
    - ExplanationFont.ttf
    - f.ttf
    - GameFont.ttf
    - InputFont.ttf
    - MathFont.ttf
  - HealthBarImage
    - HealthBarEnlarged.png
    - HealthBarGrayed.png
    - Heart.png
  - MainImage
    - Icon.png
  - MenuImage
    - Arrow.png
    - Floppy.png
    - MenuSelectTag.png
    - MenuTag.png
    - PressEnter.png
    - Settings.png
  - PlayerImage
    - Player2Picture.jpg
    - ProfilePicture.png
  - ShopImage
    - Chestplate.png
    - Leggings.png
    - Potion1.png
    - Potion2.png
    - Potion3.png
    - Potion4.png
    - Scroller.png
    - Shirt.png
    - Skills.png
    - SoldOut.png
    - Sword.png
    - Trunks.png
    - Twig.png
  - SoundEffects
    - airtone_-_resonance.wav
    - audio_hero_Revolving-Door_SIPML_C-1005.wav
    - ButtonEF.wav
    - Karstenholymoly_-_Battle_of_the_Titans.wav
    - MenuBG.wav
    - ramblinglibrarian_-_Dawn_s_Battle_(Instrumental).wav
    - stab_-_Android_Battles.wav
    - Vidian_-_the_Morning.wav
  - StatsImage
    - MenuStat1.png
  - StoryImage
    - Scene0.jpg
    - Scene1.jpg
    - Scene2.jpg
    - Scene3.jpg
    - Scene4.jpg
    - Scene5.jpg
    - Scene6.jpg
    - Scene7.jpg
    - Scene8.jpg
  - Wallpapers
    - MultiplayerWallpaper.png
    - MultiplayerWallpaper_Rare.png
    - VillageWallpaper.png
  - times.ttf

[46] The grayed out files and folders are not used by the program, but are included to form a qualified environment: there is no more time for testing after those files are removed.

Questions Database
- questions_questions.sql

Game Manual
- Game Manual.txt
- readme.txt
- Sources.txt

Game Settings
- GameSettings.txt
- ProfilePicture.png

files
- Bosses.txt
- DebugInfo.txt
- ErrorFile.txt
- PlayerData.txt
- Questions.json
- TestQuestions.json
- Guide
  - Guide.docx
  - One.txt
  - Two.txt
  - Three.txt

outdated
venv
__pycache__

ABoss.py
ABossBattle.py
AGrinding.py
ALoad.py
AMenu.py
APlayer.py
ASequence.py
AShop.py
ATextBox.py
ATutor.py
AVillage.py
GlobalVar.py
Main_AdventureMode.py
Main_Battle.py
Main_EntranceSequence.py
Main_GameSettings.py
Main_MultiplayerMode.py
Main_PracticeMode.py
mDebug.py
mDebugThread.py
mErrors.py
mMultiplayerBattle.py
mMultiplayerClient.py
mMultiplayerObjects.py
mMultiplayerPackets.py
mMultiplayerServer.py

mQuestionErrors.py
mQuestionFunctions.py
mQuestionObjects.py
mQuestionRead.py
mQuestions.py
mQuestionsWorkplace.py
mSprite.py
pyAnimation.py
pyFloating.py
pyImage.py
pyInput.py
pyMenu.py
pyMouse.py
pyText.py
pyTimer.py
The Becoming of a Mathematician.py

Appendix C. Program Source Code

1. *The Becoming of a Mathematician.py*

```
"""

The Becoming of a Mathematician.py

"""

import sys, os

# Disable
def blockPrint():
    sys.stdout = open(os.devnull, 'w')

# Restore
def enablePrint():
    sys.stdout = sys.__stdout__

# Ref: https://stackoverflow.com/questions/8391411/suppress-calls-to-print-python

print(
"""

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
The Becoming of a Mathematician

(This project is a part of the HKDSE ICT SBA. Please do not
redistribute this program without the developer's permission.)

Please read the Game Manual for information on how to play
this game. Change game settings and profile picture in the
Game Settings folder.

Please ignore the libpng warnings. They have no effect on
the program, but cannot be suppressed.

Enjoy~~
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

"""
)


blockPrint()

import pygame
import simpleaudio
from pygame.locals import *
import traceback
import mErrors

enablePrint()



def MainMenu(Screen):

    pygame.init()
    pygame.font.init()

    # Screen and Background
    EntranceSequence.EntranceSequence(Screen)

    menu = pyMenu.FrameMenu(Screen, GlobalVar.CentralClock)

    menu.AddGraphics(pygame.image.load(GlobalVar.MenuIcon))
    menu.InsertText("The Becoming of a Mathematician", Title = True)
    menu.InitiateMenu()
```

```
    ad = menu.InsertMenu("Adventure Mode")
    mu = menu.InsertMenu("Multiplayer Mode")
    pr = menu.InsertMenu("Practice Mode")
    menu.InsertText("", Enter=True)
    qu = menu.InsertMenu("Quit Game")

    displayFuncSurf = menu.DisplayFunctionSurf()

    def DisplayFunction():
        Screen.fill((255, 255, 255))
        Screen.blit(displayFuncSurf, (0, 0))

    Enter = True

    run = True
    while run:

        Screen.fill((255, 255, 255))

        for event in pygame.event.get():
            GlobalVar.Quit(event)
            counter = menu.Scrolling(event)
            if counter == pr:
                Enter = True
                PracticeMode.m_PracticeMode(Screen)
            elif counter == qu:
                run = False
            elif counter == ad:
                Enter = True
                AdventureMode.m_AdventureMode(Screen)
            elif counter == mu:
                Enter = True
                MultiplayerMode.m_MultiplayerMode(Screen)

        if Enter:

            GlobalVar.audio.FeedAudio(GlobalVar.MenuMusic)
            displayFuncSurf = menu.DisplayFunctionSurf()
            pyAnimation.FadeIn(Screen, DisplayFunction, GlobalVar.CentralClock, GlobalVar.GameFPS)
            Enter = False


        menu.Blit()
        GlobalVar.LoopBundle()

    pyAnimation.FadeOut(Screen, GlobalVar.CentralClock, GlobalVar.GameFPS)
    pygame.quit()


if __name__ == '__main__':

    # すべてはここからはじまるのだ。
    try:

        import GlobalVar
        import ATextBox
        import pyText
        import pyMenu

        import Main_AdventureMode as AdventureMode
        import Main_MultiplayerMode as MultiplayerMode
        import Main_PracticeMode as PracticeMode
        import time
        import pyAnimation
        import Main_EntranceSequence as EntranceSequence



        Size = GlobalVar.SettingDict["Resolution"]
        ScreenWidth = Size[0]
```

```
        ScreenHeight = Size[1]
        Screen = pygame.display.set_mode((ScreenWidth, ScreenHeight))
        pygame.display.set_caption("The Becoming of a Mathematician")
        MainMenu(Screen)

    except mErrors.QuestionError as e:

        print("Question.json file not found. Please reinstall the game or contact the game developer.
")

    except mErrors.BossError as e:

        print("Boss.txt file not found. Please reinstall the game or contact the game developer.")

    except GlobalVar.QuitError:

        pass

    except Exception as e:

        file = open("files/ErrorFile.txt", "w")
        traceback.print_exc(file=file)
        file.close()

        try:
            popUp = ATextBox.PopUp(Screen)
            popUp.TextBox.InsertText("The game has crashed:¥n"
                                     + str(e) + "¥n Please contact the game developer.", "center
"),
            popUp.TextBox.Finalize()
            popUp.Start()

            run = True
            while run:

                for event in pygame.event.get():
                    GlobalVar.Quit(event)
                    result = popUp.TextBox.Event(event, popUp.Offset())
                    if result:
                        run = False

                popUp.Blit()
                GlobalVar.LoopBundle()

            pyAnimation.FadeOut(Screen, GlobalVar.CentralClock, GlobalVar.GameFPS)
        except:
            pass
```

2. *GlobalVar.py*

```
"""
GlobalVar.pyaudio
"""

from pygame.locals import *

import pygame

import Main_GameSettings as GameSettings
import mQuestionRead
import time
import ABoss

import sys


bool = False
def Debug(msg):
    if bool:
        print(msg)
        time.sleep(0.5)

pygame.font.init()

GameFPS = 60

# Player Profile Picture
try:
    k = pygame.image.load("Game Settings/ProfilePicture.png")
    del k
except:
    try:
        k = pygame.image.load("Game Settings/ProfilePicture.jpg")
        del k
    except:
        ProfileImage = "assets/PlayerImage/ProfilePicture.png"
    else:
        ProfileImage = "Game Settings/ProfilePicture.jpg"
else:
    ProfileImage = "Game Settings/ProfilePicture.png"

Player2Image = "assets/PlayerImage/Player2Picture.png"

# Essentials
DefaultTextBox = pygame.image.load("assets/BattleImage/BattleTextBox.png")

# Wallpapers
DefaultVillageWallpaper = "assets/Wallpapers/VillageWallpaper.png"
DefaultMultiplayerWallpaper = "assets/Wallpapers/MultiplayerWallpaper.png"
DefaultMultiplayerWallpaperRare = "assets/Wallpapers/MultiplayerWallpaper_Rare.png"

DefaultBattleWallpaper = {
    0: "assets/BattleImage/BattleFault.jpg",
    1: "assets/BattleImage/BattleOne.jpg",
    2: "assets/BattleImage/BattleTwo.jpg",
    3: "assets/BattleImage/BattleThree.jpg"
}

DefaultGrindWallpaper = "assets/BattleImage/BattleGrind.jpg"
DefaultMultiOnline = "assets/BattleImage/BattleMultiOnline.jpg"
DefaultMultiOffline = "assets/BattleImage/BattleMultiOffline.jpg"

# Battle Related
DefaultBattleBox = DefaultTextBox
DefaultBattleNameTagStr = "assets/BattleImage/BattleNameTag.png"
DefaultBattleNameTag = pygame.image.load(DefaultBattleNameTagStr)

# Health Bar (Images are loaded in as surfaces as they are small enough.)
HeartStr = "assets/HealthBarImage/Heart.png"
```

```
HealthBarBackdropStr = "assets/HealthBarImage/HealthBarGrayed.png"
HealthBarOverlayStr = "assets/HealthBarImage/HealthBarEnlarged.png"

Heart = pygame.image.load(HeartStr)
HealthBarBackdrop = pygame.image.load(HealthBarBackdropStr)
HealthBarOverlay = pygame.image.load(HealthBarOverlayStr)

# Shop
DefaultChestplate = "assets/ShopImage/Chestplate.png"
DefaultLeggings = "assets/ShopImage/Leggings.png"
DefaultSword = "assets/ShopImage/Sword.png"
DefaultShirt = "assets/ShopImage/Shirt.png"
DefaultTrunks = "assets/ShopImage/Trunks.png"
DefaultTwig = "assets/ShopImage/Twig.png"
DefaultPotion1 = "assets/ShopImage/Potion1.png"
DefaultPotion2 = "assets/ShopImage/Potion2.png"
DefaultPotion3 = "assets/ShopImage/Potion3.png"
DefaultPotion4 = "assets/ShopImage/Potion4.png"

DefaultSkills = "assets/ShopImage/Skills.png"

DefaultShopTextboxWallpaper = DefaultTextBox
DefaultShopSoldOut = "assets/ShopImage/SoldOut.png"

# Game Statistics
DefaultMenuStatOne = "assets/StatsImage/MenuStat1.png"
DefaultMenuStatTwo = DefaultSkills
DefaultMenuStatThree = DefaultSword
DefaultMenuStatFour = DefaultPotion2

# Menus (loaded in because they are frequently used.
MenuArrow = pygame.image.load("assets/MenuImage/Arrow.png")
FloaterArrow = pygame.image.load("assets/MenuImage/PressEnter.png")
DefaultMenuBackground = pygame.image.load("assets/MenuImage/MenuTag.png")
DefaultMenuSelectBackground = pygame.image.load("assets/MenuImage/MenuSelectTag.png")

DefaultMenuScroller = MenuArrow

DefaultShopTextboxFloater = FloaterArrow
DefaultShopScroller = pygame.image.load("assets/ShopImage/Scroller.png")

MenuIcon = 'assets/MainImage/Icon.png'
# https://en.wikipedia.org/wiki/List_of_mathematical_symbols

Floppy = "assets/MenuImage/Floppy.png"
Settings = "assets/MenuImage/Settings.png"

# Multiplayer
DefaultLogBox = DefaultTextBox

# Fonts
GameFont = "assets/Fonts/GameFont.ttf"
BlockyFont = "assets/Fonts/BlockyFont.ttf"
MathFont = "assets/Fonts/MathFont.ttf"
ExplanationFont = "assets/Fonts/ExplanationFont.ttf"
InputFont = "assets/Fonts/InputFont.ttf"
MathInputFont = InputFont

BattleFontString = GameFont

# Pre-loaded
FONTHEIGHT = 100
BattleFont = pygame.font.Font(BlockyFont, FONTHEIGHT)
DefaultShopTextboxTextFontString = GameFont
DefaultShopTextboxInputFontString = InputFont
DefaultShopTitleFontString = GameFont
DefaultShopBalanceFontString = GameFont
DefaultShopFontString = GameFont
DefaultInputFontString = InputFont
DefaultMenuTitleFontString = GameFont
DefaultMenuHeaderFontString = GameFont
```

```
DefaultMenuTextFontString = GameFont
DefaultMenuExplanationFontString = ExplanationFont
DefaultMenuMenuFontString = GameFont

# Files
QuestionDatabase = mQuestionRead.ReadQuestion("files/Questions.json")
BossDatabase = ABoss.ReadBoss("files/Bosses.txt")
PlayerDataFileName = "files/PlayerData.txt"
SettingDict = GameSettings.GameSettings("Game Settings/GameSettings.txt")
ErrorFileName = "files/ErrorFile.txt"
DebugFile = "files/DebugInfo.txt"
GuideFilename = ["files/Guide/One.txt", "files/Guide/Two.txt", "files/Guide/Three.txt"]
GuidePath = "files/Guide"

LevelNames = {
    1 : "Quadratic Equations and Graphs",
    2 : "Arithmetic Sequence",
    3 : "Geomtrtic Sequences and Polynomials"
}

PlayerHP = 100000
AttackDmg = {1: 15000, 2: 16500, 3: 18000}

# Battle
DefendPotionPerc = 0.05
AttackPotionPerc = 0.05
HealthPotionPerc = 0.03

# Global
CentralClock = pygame.time.Clock()

def LoopBundle():
    pygame.display.update()
    CentralClock.tick(30)

class QuitError(Exception):
    pass

def Quit(event):
    if event.type == QUIT:
        pygame.quit()
        raise QuitError

class Color:

    def __init__(self):
        self.black = (0, 0, 0)
        self.white = (255, 255, 255)
        self.red = (255, 0, 0)
        self.blue = (0, 0, 255)

color = Color()

PlayMusic = True


DefaultPort = 40000

# Sound effects
import simpleaudio
ButtonClick = simpleaudio.WaveObject.from_wave_file("assets/SoundEffects/ButtonEF.wav")

BattleMusic1 = "assets/SoundEffects/Karstenholymoly_-_Battle_of_the_Titans.wav"
BattleMusic2 = "assets/SoundEffects/ramblinglibrarian_-_Dawn_s_Battle_(Instrumental).wav"
BattleMusic3 = "assets/SoundEffects/stab_-_Android_Battles.wav"
BattleMusic = [BattleMusic1, BattleMusic2, BattleMusic3]
MenuMusic = "assets/SoundEffects/MenuBG.wav"
VillageMusic = "assets/SoundEffects/Vidian_-_the_Morning.wav"
AmbientMusic = "assets/SoundEffects/airtone_-_resonance.wav"
ShopMusic = "assets/SoundEffects/audio_hero_Revolving-Door_SIPML_C-1005.wav"
```

```
import wave
import pyaudio
import threading

PYAUDIO = pyaudio.PyAudio()

class AudioClass(threading.Thread) :

    CHUNK = 44100

    def __init__(self):
        super(AudioClass, self).__init__(daemon=True)
        self.AUDIO = None
        self.replayAudio = None

    def FeedAudio(self, aud):
        self.AUDIO = aud
        self.replayAudio = aud

    def run(self):
        while True:
            if self.AUDIO:
                fileobj = wave.open(self.AUDIO, 'rb')
                self.AUDIO = None

                stream = PYAUDIO.open(format = PYAUDIO.get_format_from_width(fileobj.getsampwidth
()),
                                      channels = fileobj.getnchannels(),
                                      rate = fileobj.getframerate(),
                                      output = True)

                data = fileobj.readframes(self.CHUNK)
                while data and not self.AUDIO:
                    stream.write(data)
                    data = fileobj.readframes(self.CHUNK)

                stream.stop_stream()
                del stream
                del fileobj

                self.AUDIO = self.replayAudio


# A version that is not working and the reason may be the following:
# OSError: [Errno -9985] Device unavailable
# The accidental looping of generating a stream causes the
# device to dysfunction
"""
def run(self):
    while True:
        if self.AUDIO:
            # Load
            fileobj = wave.open(self.AUDIO, 'rb')
            stream = PYAUDIO.open(format=PYAUDIO.get_format_from_width(fileobj.getsampwidth()),
                                  channels=fileobj.getnchannels(),
                                  rate=fileobj.getframerate(),
                                  output=True)
            # Play
            data = fileobj.readframes(self.CHUNKS)
            while data != "" and [[[not self.AUDIO]]]:
                stream.write(data)
                data = fileobj.readframes(self.CHUNKS)
"""

audio = AudioClass()
if SettingDict["Music"]:
    audio.start()
```

3. *mSprite.py*

```
"""
mSprite.py
"""

import pyImage

class Sprite:

    __slots__ = "name", "fullHP", "graphics", "nowHP"

    def __init__(self, name, fullHP, graphics, nowHP = -1):

        self.name = name
        self.fullHP = fullHP
        self.graphics = graphics
        self.nowHP = self.fullHP if nowHP == -1 else nowHP

    def returnSurface(self, height):
        return pyImage.ScaleGraphics(self.graphics, height, False, True)

    def copy(self):
        return Sprite(self.name, self.fullHP, self.graphics)
```

4. *Main_AdventureMode.py*

```
'''
Main_AdventureMode.py
'''

import ALoad
import AVillage
import ASequence

def m_AdventureMode(screen):

    playerInfo, tutorialSequence, cont = ALoad.LoadOptions(screen)

    if cont:
        # Tutorial Sequence
        if tutorialSequence:
            ASequence.AdventureModeSequence(screen)

        # Start the game
        AVillage.a_Village(screen, playerInfo, tutorialSequence)
```

5. *Main_EntranceSequence.py*

```
"""
Main_EntranceSequence.py
"""

import pygame
import pyAnimation
import pyText
import GlobalVar
import ATextBox
from pygame.locals import *


def EntranceSequence(screen):

    width = screen.get_width()
    height = screen.get_height()

    font = pygame.font.Font(GlobalVar.BlockyFont, height//10)
    lines = pyText.TextWrap("The Becoming of¥na Mathematician", font, int(width*4/5))
    displaySurf = pyText.TextRender(lines, font, int(height/20), "center", (255, 255, 255))
    rect = displaySurf.get_rect()
    rect.center = (width//2, height//2)

    def DisplayFunction():
        screen.fill((0, 0, 0))
        screen.blit(displaySurf, rect)

    pyAnimation.FadeIn(screen, DisplayFunction, GlobalVar.CentralClock, GlobalVar.GameFPS)

    popUp = ATextBox.PopUp(screen)

    def Menu(msg):

        def func():
            popUp.TextBox.InsertText(msg,
                                     "center", bold = True)
            popUp.TextBox.InsertMenu("OK.")
            popUp.TextBox.InsertMenu("Understood.")

        return func

    functionList = [
        lambda : popUp.TextBox.InsertText("When you see an arrow on the bottom right corner, "
                                          "press Enter to proceed.", "center", bold = True),
        Menu("In a menu as shown, use the up, down buttons to choose an option."),
        Menu("You can also use your mouse to click on the options."),
        lambda: popUp.TextBox.InsertText("Please read the game manual for more information.", "cent
er", bold=True),
        lambda : popUp.TextBox.InsertText("Have fun!¥n(You may skip this tutorial by pressing "
                                          "ESC next time.)", "center", bold = True)
    ]

    DisplayFunction()
    pygame.time.delay(1000)

    for i in functionList:

        popUp.TextBox.Reset()
        i()
        popUp.TextBox.Finalize()

        DisplayFunction()
        popUp.Start()

        run = True
        while run:

            for event in pygame.event.get():
                GlobalVar.Quit(event)
```

```
            if event.type == KEYDOWN:
                if event.key == K_ESCAPE:
                    pyAnimation.FadeOut(screen, GlobalVar.CentralClock, GlobalVar.GameFPS)
                    return
            result = popUp.TextBox.Event(event, popUp.Offset())
            if result or result == 0:
                run = False

        popUp.Blit()
        GlobalVar.LoopBundle()

pyAnimation.FadeOut(screen, GlobalVar.CentralClock, GlobalVar.GameFPS)
```

6. *Main_GameSettings.py*

```
"""
Main_GameSettings.py
"""

import re

def GameSettings(filename):

    try:
        file = open(filename, "r")
    except:
        open(filename, "w")
    else:
        lines = file.readlines()

        settingDict = {
            'Resolution' : "(1200,800)",
            'Unicode' : '1',
            'Debug' : '0',
            'EasyMode' : '0',
            'Music' : '1'
        }

        for line in lines:

            if '#' in line:
                continue
            else:
                line = line.replace(" ", "")
                segments = re.split("[:¥n]", line)
                if segments[0] in settingDict.keys():
                    settingDict[segments[0]] = segments[1]

        ResolutionList = re.split("[()¥]¥[,]", settingDict["Resolution"])
        tempList = []
        for i in ResolutionList:
            if i:
                tempList.append(int(i))
        settingDict["Resolution"] = tempList

        def Set(item):
            settingDict[item] = True if settingDict[item].lower() in ["true", "1"] else False

        Set("Unicode")
        Set('Debug')
        Set('EasyMode')
        Set('Music')

        file.close()
        return settingDict
```

7. *Main_MultiplayerMode.py*

```
"""
Main_MultiplayerMode.py
"""

import re
import time

import Main_Battle as Battle
import pyMenu
import pygame
import GlobalVar
import pyAnimation
import mMultiplayerBattle
import mMultiplayerServer
import socket
import random

import multiprocessing
multiprocessing.freeze_support()

class ConnectionError(Exception):
    pass


# Tentative IP Test Plan:
"""

a Valid Data (Random)
b Extreme Data
c Illegal Data

OpenServer
1. (a) {Open a server with port 40000}
2. (a) 40000 {Open a server with port 40000}
3. (a) 80 {Open a server with port 80 (non-deterministic, since <1024 is reserved)}
4. (b) 1 {""}
5. (b) 65535 {""}
6. (c) 0, 65536 [Invalid Value] {Message appears}
7. (c) 100000 [Invalid Value] {Message appears}
8. (c) HAHA [Invalid Data Type] {No reaction}

ConnectOptions -> Manual Connection
1-5. (a, b) 192.168.1.74 & Port Number in previous test
{1-5: Connects normally (except for 3, 4)}
6,7. (c) ""
{Invalid Port Number message}
8. (c) HAHA
{Invalid IP Format}
9. (c) 192.168.1.
{Invalid IP Format}
10. (c) 333.333.333.333
{Invalid IP Format}
11. (c) 192.168.1.74192.168.1.74
{Invalid IP Format}
12. (c) 192.168.1.74:40000192.168.1.74:40000
{Invalid IP Format}
13. (c) 192.168.1.74:40000 192.168.1.74:40000
{Connects like 1}
14. (c) 192.168.1.74:40000d 192.168.1.74:40000
{Connects like 1}
15. (c) 192.168.1.74:
{Invalid IP Format}



"""
```

```
def m_MultiplayerMode(screen):

    ip_address = socket.gethostbyname(socket.gethostname())
    myPort = GlobalVar.DefaultPort

    serverThread = multiprocessing.Process(target=mMultiplayerServer.mainfunc, args=(ip_address,))

    def OpenServer(screen):

        menu = pyMenu.FrameMenu(screen)

        menu.InsertText("Online Multiplayer Battle", Title=True)
        menu.InsertText("Please enter a port number to open your server on.", Header=True)
        menu.InsertText("(Type nothing to set default.)", Header=True)

        menu.InsertText("", Enter=True)

        input = menu.InsertInput(">> ", inputColor=pyMenu.Color("blue"), Header=True)

        menu.InsertText("", Enter=True)

        failed = False

        confirmed = False
        while not confirmed:
            screen.fill(pygame.Color("white"))
            menu.Blit()
            menu.UpdateCursor()
            for event in pygame.event.get():
                GlobalVar.Quit(event)
                counter = menu.Scrolling(event)
                menu.Typing(event)
                if counter == input:
                    theText = menu.Text()
                    try:
                        portNo = int(theText)
                        if 0 < portNo <= 65535:
                            return portNo
                        else:
                            if not failed:
                                failed = True
                                menu.InsertText("Accepted Range: 0 < Port Number <= 65535", Text=True,
color=(255, 0, 0))
                            raise ConnectionError("Invalid Port Number.")
                    except:
                        if not theText:
                            return GlobalVar.DefaultPort
                        else:
                            pass
            GlobalVar.LoopBundle()

    def IPAddress(screenSurf):

        screen = pyMenu.FrameMenu(screenSurf)

        screen.InsertText("Online Multiplayer Battle", Title=True)
        screen.InsertText("Please enter an IP address to connect to.", Header=True)

        screen.InsertText("", Enter=True)

        input = screen.InsertInput(">> ", inputColor=pyMenu.Color("blue"), Header=True)

        screen.InsertText("", Enter=True)

        screen.InitiateMenu()
        rtn = screen.InsertMenu("Return")

        confirmed = False
        while not confirmed:
            screenSurf.fill(pygame.Color("white"))
            screen.Blit()
```

```
            screen.UpdateCursor()
            for event in pygame.event.get():
                GlobalVar.Quit(event)
                counter = screen.Scrolling(event)
                screen.Typing(event)
                if counter == rtn:
                    return None
                elif counter == input:
                    try:
                        userInput = screen.Text()
                        if re.fullmatch("¥d+¥.¥d+¥.¥d+¥.¥d+:¥d+|¥d+¥.¥d+¥.¥d+¥.¥d+", userInput):
                            pass
                        else:
                            raise ConnectionError("Invalid IP Format. Format: XX.XX.XX.XX:PortNo."
                                                  " Do not use hostnames.")
                        ipList = re.split(":", userInput)
                        no = re.split('¥.', ipList[0])
                        for i in no:
                            if not 0 <= int(i) <= 255:
                                raise ConnectionError("Invalid IP Format. Octets must lie between 0 to
255.")
                        if len(ipList) == 1:
                            ip = ipList[0]
                            port = GlobalVar.DefaultPort
                        elif len(ipList) == 2:
                            ip = ipList[0]
                            port = int(ipList[1])
                            if not 0 < port <= 65535:
                                raise ConnectionError("Invalid Port Number.")
                        return ip, port
                    except ConnectionError as e:
                        raise e
                    except Exception as e:
                        raise ConnectionError("Invalid IP Format: " + str(e))
            GlobalVar.LoopBundle()



    def Name(screen):
        menu = pyMenu.FrameMenu(screen, GlobalVar.CentralClock)

        menu.InsertText("Online Multiplayer Battle", Title=True)
        menu.InsertText("Please enter your name.", Header=True)
        menu.InitiateMenu()

        menu.InsertText("", Enter=True)

        input = menu.InsertInput(">> ", inputColor=pyMenu.Color("blue"), Header=True)

        menu.InsertText("", Enter=True)

        rtn = menu.InsertMenu("Return")

        confirmed = False
        while not confirmed:
            screen.fill(pygame.Color("white"))
            menu.Blit()
            menu.UpdateCursor()
            for event in pygame.event.get():
                GlobalVar.Quit(event)
                counter = menu.Scrolling(event)
                menu.Typing(event)
                if counter == rtn:
                    confirmed = False
                    return
                elif counter == input:
                    if menu.Text():
                        if len(menu.Text()) > 16:
                            return menu.Text()[:16]
                        else:
                            return menu.Text()
```

```
                    confirmed = True
            GlobalVar.LoopBundle()

    def ConnectionFailed(screen, e):

        menu = pyMenu.FrameMenu(screen, GlobalVar.CentralClock)

        menu.InsertText("Online Multiplayer Battle", Title=True)
        menu.InsertText("Connection has failed! ", Header=True)
        menu.InsertText("Please check if the server has been turned on at that IP address, ", Text=T
rue)
        menu.InsertText("or if the IP address you entered is valid.", Text=True)

        menu.TextWrap("Error code: " + str(e), Text=True, color=pygame.Color("gray30"))

        menu.InsertText("", Enter=True)

        menu.InitiateMenu()
        menu.InsertMenu("Troubleshoot")
        menu.InsertMenu("Return")

        confirmed = False
        while not confirmed:
            screen.fill(pygame.Color("white"))
            menu.Blit()
            for event in pygame.event.get():
                GlobalVar.Quit(event)
                counter = menu.Scrolling(event)
                if counter == 1:
                    confirmed = True
                    return None
                elif counter == 0:
                    TroubleShoot(screen)

            GlobalVar.LoopBundle()

    def OfflineMulti(screen):

        battle = Battle.OfflineBattle(screen)

        pyAnimation.EnterFade(screen, battle.Fade, GlobalVar.CentralClock, GlobalVar.GameFPS)

        battle.Main()

    def ConnectOption(screen):

        menu = pyMenu.FrameMenu(screen, GlobalVar.CentralClock)

        menu.InsertText("Online Multiplayer Battle", Title=True)
        menu.InsertText("Please select a connection option.", Header=True)

        menu.InsertText("", Enter=True)
        menu.InitiateMenu()

        doAuto = menu.InsertMenu("Open a Server")
        menu.InsertText("Enter the server that you have opened (in the Multiplayer "
                        "Mode menu). It may fail if you enter too soon.",
                        Explanation=True)

        doIp = menu.InsertMenu("Manual Connection")
        menu.InsertText("Manually input an IP address. (Please ensure the server is up and running)",
 Explanation=True)

        menu.InsertText("", Enter=True)

        rtn = menu.InsertMenu("Return")

        confirmed = False
        while not confirmed:
            screen.fill(pygame.Color("white"))
            menu.Blit()
```

```
            for event in pygame.event.get():
                GlobalVar.Quit(event)
                counter = menu.Scrolling(event)
                if counter == rtn:
                    confirmed = False
                    return None
                elif counter == doIp:
                    confirmed = False
                    return IPAddress(screen)
                elif counter == doAuto:
                    confirmed = False
                    return ip_address, myPort

            GlobalVar.LoopBundle()



    def OnlineMulti(screen):

        name = Name(screen)
        if name == None:
            return
        returnList = ConnectOption(screen)
        if not returnList:
            return
        else:
            ip, port = returnList
        m = mMultiplayerBattle.OnlineBattle(screen = screen, IP = ip, port = port, name = name)
        m.Main()

    def TroubleShoot(screen):

        menu = pyMenu.FrameMenu(screen)

        menu.InsertText("Connection Errors", Title=True)

        menu.InsertText("[Errno 10057] ...", Text=True)
        menu.TextWrap("I told you to wait... But if the problem persists, please contact "
                      "the game developer as this is an unsolved issue.", Explanation=True)
        menu.InsertText("", Explanation=True)

        menu.InsertText("[Errno 11004] getaddrinfo failed", Text=True)
        menu.TextWrap("The IP address you have entered is invalid. Please enter IP"
                      + " addresses, not host names (hostnames have nondeterministic behavior).", Expl
anation=True)
        menu.InsertText("", Explanation=True)

        menu.TextWrap("[WinError 10054] An existing connection was forcibly closed by the remote hos
t", Text=True)
        menu.TextWrap("The server has closed. Consult the server host.", Explanation=True)
        menu.InsertText("", Explanation=True)

        menu.InsertText("Timed out", Text=True)
        menu.TextWrap("It may happen because the connection is bad. "
                      + "If the error repeatedly occurs, please contact the game developer.", Explanat
ion=True)

        menu.InsertText("", Enter=True)

        menu.InitiateMenu()
        menu.InsertMenu("Return")

        confirmed = False
        while not confirmed:

            screen.fill(pygame.Color("white"))
            menu.Blit()
            for e in pygame.event.get():
                counter = menu.Scrolling(e)
                if counter == 0:
                    return
```

```
        GlobalVar.LoopBundle()

    # ************************************************************************************************
    # ********************End of subprogram definition*********************************************
    # ************************************************************************************************

    mainMenu = pyMenu.FrameMenu(screen, GlobalVar.CentralClock)

    # Wallpaper
    if random.random() > 0.95:
        mainMenu.AddGraphics(GlobalVar.DefaultMultiplayerWallpaperRare)
    else:
        mainMenu.AddGraphics(GlobalVar.DefaultMultiplayerWallpaper)

    mainMenu.InsertText("Multiplayer Mode", Title = True)

    mainMenu.InitiateMenu()
    mainMenu.InsertMenu("Offline Game")
    mainMenu.InsertMenu("Online Game")
    mainMenu.TextWrap("For insturctions on the online Multiplayer Mode, please read the "
                    "¥"Multiplayer Manual¥" in the ¥"Game Settings¥" Folder", Text=True)

    mainMenu.InsertText("", Text=True)
    sava = mainMenu.InsertMenu("Open a Server")
    mainMenu.TextWrap("Please wait around 1 minute for the server to start up. "
                    "A message will appear in the console when the server is ready.", Text=True)

    mainMenu.InsertText("", Text=True)
    exi = mainMenu.InsertMenu("Return to Main Menu")

    def displayFunction():

        screen.fill(GlobalVar.color.white)
        mainMenu.Blit()

    displayFunctionSurf = mainMenu.DisplayFunctionSurf()
    def display2Function():
        screen.blit(displayFunctionSurf, (0, 0))

    pyAnimation.EnterFade(screen, display2Function, GlobalVar.CentralClock, GlobalVar.GameFPS)

    exitFlag = False

    while not exitFlag:

        displayFunction()

        for event in pygame.event.get():
            GlobalVar.Quit(event)

            counter = mainMenu.Scrolling(event)

            if counter == 0:
                OfflineMulti(screen)
                GlobalVar.audio.FeedAudio(GlobalVar.MenuMusic)
            elif counter == 1:
                try:
                    OnlineMulti(screen)
                    GlobalVar.audio.FeedAudio(GlobalVar.MenuMusic)
                except Exception as e:
                    ConnectionFailed(screen, e)
            elif counter == sava:
                try:
                    if not serverThread.is_alive():
                        port = OpenServer(screen)
                        myPort = port
                        serverThread = multiprocessing.Process(target=mMultiplayerServer.mainfunc,
                                                               args=(ip_address, port))
```

```
                    serverThread.start()
                except:
                    pass
            elif counter == exi:
                exitFlag = True

            if counter not in [None, sava]:
                pyAnimation.FadeOut(screen, GlobalVar.CentralClock, GlobalVar.GameFPS)

        GlobalVar.LoopBundle()

    if serverThread.is_alive():
        try:
            serverThread.terminate()
        except:
            pass


if __name__ == '__main__':

    import warnings

    warnings.filterwarnings('error', message='libpng warning: iCCP: known incorrect sRGB profile.*
')

    m_MultiplayerMode(pygame.display.set_mode((int(1280//1.5), int(800//1.5))))
```

8. *Main_PracticeMode.py*

```
"""
Main_PracticeMode.py
"""


import pyText
import pyInput
import pyMenu
import mQuestionRead
import mQuestions
import pygame
from pygame.locals import *
import GlobalVar

def m_PracticeMode(screen):

    menu = pyMenu.FrameMenu(screen)

    menu.InsertInput("Difficulty (1-3): ", Text=True, inputColor = (255, 0, 0))

    run = True
    while run:

        confirmed = menu.Typing()
        if confirmed:
            try:
                diff = int(menu.Text())
                if 1 <= diff <= 3:
                    run = False
            except:
                pass

        screen.fill((255, 255, 255))
        menu.Blit()
        GlobalVar.LoopBundle()

    # Stage 2 : Display question and ask for answer
    questionObj = mQuestions.QuestionGenerator(diff, GlobalVar.QuestionDatabase)
    question = questionObj["question"]
    answer = questionObj["answer"]
    time = questionObj["time"]
    diffPt = questionObj["diffPt"]


    menu = pyMenu.FrameMenu(screen)
    menu.InsertText("Difficulty (1-3):    " + str(diff), Text=True)
    menu.InsertText("", Enter=True)
    menu.InsertText("Question:", Subheader=True)
    menu.InsertText(f"Time Limit: {str(time):10}"
                    + f"Difficulty Variation: {str(diffPt):10}", Explanation=True)

    menu.TextWrap(question, Text=True)

    menu.InsertInput("Input your answer: ", inputColor=(255,0,0), Text=True)

    run = True
    while run:


        confirmed = menu.Typing()
        if confirmed:
            ans = menu.Text()
            run = False

        screen.fill((255, 255, 255))
        menu.Blit()
        GlobalVar.LoopBundle()

    correct = mQuestions.AnswerChecker(answer, ans)
```

```
    # Stage 3: Displaying results and asking if the player wants to play again

    menu = pyMenu.FrameMenu(screen)
    menu.InsertText("Difficulty (1-3):    " + str(diff), Text=True)
    menu.InsertText("", Enter=True)
    menu.InsertText("Question:", Subheader=True)
    menu.InsertText(f"Time Limit: {str(time):10}"
                        + f"Difficulty Variation: {str(diffPt):10}", Explanation=True)

    menu.TextWrap(question, Text=True)

    menu.InsertText("Input your answer:    " + ans, Text=True)
    menu.InsertText(("You are correct!" if correct else "You are incorrect!") + " Correct answer: "
 + str(answer), Text=True)

    menu.InsertText("", Enter=True)
    menu.InsertText("Would you like to play again?", Subheader=True)
    menu.InitiateMenu()
    ya = menu.InsertMenu("Yes")
    nah = menu.InsertMenu("Nah")

    run = True
    while run:

        for event in pygame.event.get():
            GlobalVar.Quit(event)
            ha = menu.Scrolling(event)
            if ha == ya:
                m_PracticeMode(screen)
                return
            elif ha == nah:
                return

        screen.fill((255, 255, 255))
        menu.Blit()
        GlobalVar.LoopBundle()



if __name__ == '__main__':
    m_PracticeMode(pygame.display.set_mode((1280, 800)))
```

9. *Main_Battle.py*

```
"""
Main_Battle.py
"""

import pygame

import mDebug
import pyText
from mSprite import *
import pyMenu
import pyInput
import pyTimer
from pygame.locals import *
from pyImage import *
import mQuestions
import pyFloating
import random
import math
from GlobalVar import *


pygame.init()
pygame.font.init()


class Battle:

    # Battle is a class because we would like to use
    # inheritance to add sequences and edit the "main" function
    # The implementation for the Adventure Mode grinding,
    # monster battles and multiplayer mode are too similar
    # to not use inheritance with

    # Constants
    BoxShift = 0.05 # Box Text shift is 0.07 height
    BoxSpacing = 0.02 # Box Text spacing is 0.02 height
    DisplayHeight = 0.6 # Graphics take 40% of the screen height
    BoxHeight = 1 - DisplayHeight
    TextShift = 0.03 # Distance between special objects and text
    TextSize = 0.03
    FloatWidth = 0.07
    FloatHeight = 0.06

    SelectionWidth = 0.6


    # Initiator Function
    def __init__(self, screen, diff, playerSprite, oppSprite,
                 displayWallpaper = None,
                 textBoxFrame = None,
                 LightName = False
                 ):

        # Copying Arguments
        self.screen = screen
        self.height = screen.get_height()
        self.width = screen.get_width()
        self.diff = diff

        # Display Constants
        self.boxShift = int(self.BoxShift * self.height)
        self.boxSpacing = int(self.BoxSpacing * self.height)
        self.displayHeight = int(self.DisplayHeight * self.height)
        self.boxHeight = self.height - self.displayHeight
        self.textShift = int(self.TextShift * self.width)
        self.textSize = int(self.TextSize * self.height)

        # Floater (The arrow that prompts the player to press Enter)
        self.floatWidth = int(self.FloatWidth * self.width)
```

```
        self.floatHeight = int(self.FloatHeight * self.height)
        self.floaterRect = (self.width - self.floatWidth - self.boxShift * 2,
                            self.height - self.floatHeight - self.boxShift)
        self.floater = pyFloating.Floating(FloaterArrow, (self.floatWidth, self.floatHeight), Centr
alClock)

        # Fonts
        self.MathFont = pygame.font.Font(MathFont, self.textSize)
        self.MathInputFont = pygame.font.Font(MathInputFont, self.textSize)
        self.BattleFont = pygame.font.Font(BattleFontString, self.textSize)
        self.BoldBattleFont = pygame.font.Font(BattleFontString, self.textSize)
        self.BoldBattleFont.set_bold(True)

        # Game state
        self.playerSprite = playerSprite.copy()
        self.oppSprite = oppSprite.copy()
        self.round = 0
        self.playerBool = True # Whether the player is alive
        self.oppBool = True # Whether the opponent is alive

        # Selection Menus
        self.selectionHeight = self.textSize
        self.selectionWidth = int(self.SelectionWidth * self.width)
        self.selectionSurface = (self.selectionWidth, self.selectionHeight)

        # Display options
        if displayWallpaper == None:
            wallpaper = pygame.image.load(DefaultBattleWallpaper[0])
        else:
            wallpaper = pygame.image.load(displayWallpaper)

        wallpaper = ScaleGraphics(wallpaper, self.height, False, True)
        if wallpaper.get_width() < self.width:
            wallpaper = pygame.transform.scale(wallpaper, (self.width, self.height))
        width = wallpaper.get_width()
        self.displayWallpaper = Crop(wallpaper, (int((width - self.width) / 2), 0), (self.width, se
lf.displayHeight))
        self.textBoxWallpaper = Crop(wallpaper, (int((width - self.width) / 2), self.displayHeight),
 (self.width, self.boxHeight))

        if textBoxFrame == None:
            battleBox = DefaultBattleBox
            self.textBoxFrame = pygame.transform.scale(battleBox, (self.width, self.boxHeight))
        else:
            self.textBoxFrame = pygame.transform.scale(pygame.image.load(textBoxFrame), (self.width,
 self.boxHeight))

        self.initSurf = self.FadeSurf()
        self.LightName = LightName

        audio.FeedAudio(BattleMusic[random.randint(0, 2)])

    # Execution Sequence
    def Main(self):
        # Generate a question
        questionObj = mQuestions.QuestionGenerator(self.diff, QuestionDatabase)
        self.RefreshBattle()
        self.DisplayQuestion(questionObj)

        self.RefreshBattle()
        self.DealDamage("player", 2000)
        self.RefreshBattle()
        self.DealDamage("opponent", 300)

    # Main Display Function: Displays the background
    def RefreshBattle(self):

        self.screen.fill(color.white)
        self.screen.blit(self.displayWallpaper, (0, 0))
        if self.textBoxWallpaper != None:
            self.screen.blit(self.textBoxWallpaper, (0, self.displayHeight))
```

```
        self.screen.blit(BattleDisplay(self.playerSprite, self.oppSprite,
                    (self.width, self.displayHeight),
                    round = self.round, playerBool = self.playerBool, oppBool = self.oppBool, LightN
ame = self.LightName
                                    ), (0, 0))

    def DisplayTextBox(self):

        self.screen.blit(self.textBoxFrame, (0, self.displayHeight))

    def FadeSurf(self):

        tempSurf = pygame.Surface(self.screen.get_size(), pygame.SRCALPHA)
        tempSurf.fill(color.white)
        tempSurf.blit(self.displayWallpaper, (0, 0))
        if self.textBoxWallpaper != None:
            tempSurf.blit(self.textBoxWallpaper, (0, self.displayHeight))
        tempSurf.blit(BattleDisplay(self.playerSprite, self.oppSprite,
                                    (self.width, self.displayHeight),
                                    round=self.round, playerBool=self.playerBool, oppBool=self.oppBo
ol), (0, 0))
        tempSurf.blit(self.textBoxFrame, (0, self.displayHeight))
        return tempSurf

    def Fade(self):

        self.screen.blit(self.initSurf, (0, 0))

    def LoopBundle(self):
        pass

    # Displays floater
    def DisplayFloater(self):
        self.screen.blit(self.floater.Update(), self.floaterRect)

    # The following are sequences in the game.
    # Sequences are in series.

    # DisplayQuestion displays the Q&A part of the sequence
    # and returns "result" and "timeLeft" for damage calculation
    # it is a dictionary
    def DisplayQuestion(self, questionObj):

        # Sequences in parallel
        def Display(stage):

            if stage >= 1:

                # Displays the background
                self.DisplayTextBox()

                # Displays the timer
                topCoord = self.displayHeight + self.boxShift
                timerSurf = timer.Surface()
                self.screen.blit(timerSurf, (self.width - self.boxShift * 2 - timerWidth, topCoord))

                # Display the timeLeft message
                topCoord += int((int(self.textSize * 1.5) - self.textSize) / 2)
                self.screen.blit(timeLeftSurf,
                                 (self.width - self.boxShift * 2 - timerWidth - self.textShift - timeLef
tSurf.get_width(),
                                                 topCoord))

                # Displays the question label
                topCoord += int((int(self.textSize * 1.5) - self.textSize) / 2)
                self.screen.blit(questionLabel, (self.boxShift, topCoord))

                # Displays the question
                topCoord += self.boxSpacing + self.textSize
                self.screen.blit(questionSurf, (self.boxShift, topCoord))
```

```
            # Displays the input box
            topCoord += questionHeight + self.boxSpacing
            self.screen.blit(answerPromptSurf, (self.boxShift, topCoord))
            self.screen.blit(questionInput.Surface(), (self.boxShift + answerPromptSurfWidth + s
elf.textShift, topCoord))

            if stage >= 2:

                # Displays the results
                topCoord += answerPromptSurfHeight + self.boxSpacing
                self.screen.blit(endSurf, (self.boxShift, topCoord))

                # Displays the Enter arrow
                self.DisplayFloater()

        LoopBundle()

        pass

    question = questionObj["question"]
    answer = questionObj["answer"]
    time = int(questionObj["time"])

    # Readies the player to answer the question
    startPrompt = pyText.TextRender(
        pyText.TextWrap(
            "You are about to answer a question.¥n" + "Time limit : " + str(time) + "¥n Press Ente
r to continue",
            self.BattleFont, int(self.width * 3 / 4)
        ),
        self.BattleFont, self.boxSpacing, "center", color = color.red
    )
    startRect = startPrompt.get_rect()
    startRect.top = self.displayHeight + self.boxShift
    startRect.centerx = int(self.width // 2)

    # Displays the pre-question prompt
    confirmed = False
    while not confirmed:

        if self.LoopBundle():
            break

        self.DisplayTextBox()
        self.screen.blit(startPrompt, startRect)
        self.DisplayFloater()
        pygame.display.update()

        for event in pygame.event.get():
            Quit(event)
            if event.type == KEYDOWN:
                if event.key == K_RETURN:
                    confirmed = True
                    break

        LoopBundle()

    # Renders the question text
    questionLabel = self.BattleFont.render("Question", 1, color.black)
    questionText = pyText.TextWrap(mDebug.QuestionType(questionObj["comment"]) + question,
                                   self.MathFont, self.width - 2 * self.boxShift)
    questionSurf = pyText.TextRender(questionText, self.MathFont, int(self.boxSpacing//3))
    questionHeight = questionSurf.get_height()

    # Creates answer prompt
    answerPromptSurf = self.BattleFont.render("Input your answer: ", 1, color.black)
    answerPromptSurfWidth = answerPromptSurf.get_width()
    answerPromptSurfHeight = answerPromptSurf.get_height()
    questionInput = pyInput.TextInput(self.MathInputFont,
                                      self.width - 2 * self.boxShift - answerPromptSurfWidth - self.t
extShift, CentralClock)
```

```
        # Creates a timer
        timer = pyTimer.Timer(time, CentralClock, int(self.textSize * 1.5), color=color.red)
        timeLeftSurf = self.BattleFont.render("Time Left:", 1, color.black)
        timerWidth = pygame.font.Font(BattleFontString, int(self.textSize * 1.5)).size("00:00.0")
[0]

        # Stage 1: Question Sequence
        while timer.Update() and not questionInput.Typing():
            if self.LoopBundle():
                break
            Display(1)

        # Assigning results
        timeLeft = timer.Time()
        userInput = questionInput.Text()

        # If-else states to determine the end-game message
        if timeLeft == 0:
            result = False
            endMessage = "Time's up!"
        else:
            result = mQuestions.AnswerChecker(answer, userInput)
            if result:
                endMessage = "You are correct!"
            else:
                endMessage = "You are incorrect!"

        endMessage += " Correct Answer: " + (f"{answer:.3g}" if type(answer) == type(1.1) else str(a
nswer))
        endSurf = self.BattleFont.render(endMessage, 1, color.blue)

        # Stage 2: Displaying results
        confirmed = False
        while not confirmed:
            if self.LoopBundle():
                break
            Display(2)

            for event in pygame.event.get():
                Quit(event)
                if event.type == KEYDOWN:
                    if event.key == K_RETURN:
                        confirmed = True

        mDebug.SaveState(questionObj, userInput, result, questionObj["time"] - timeLeft)

        return {
            "result" : result,
            "timeLeft" : timeLeft
                }

    # DealDamage displays the damage dealt.
    # Player deals damage when side = "player"
    # Opponent deals damage when side = "opponent"
    def DealDamage(self, side, damage, dodgeFlag = False):

        # Is the ___ dead?
        # Creates the damage message
        if side == "player":
            self.oppSprite.nowHP -= damage
            if self.oppSprite.nowHP <= 0:
                self.oppSprite.nowHP = 0
                self.oppBool = False
            dmgMessage = pyText.TextWrap(
                "You have" + " dealt " +
                ((str(damage) + " damage") if damage else "no damage") + " on " +
                 self.oppSprite.name +
                (
                    ("¥n¥n" + (self.oppSprite.name + " has died!"))
                    if not self.oppBool else ""
```

```
            ),
            self.BattleFont, int((self.width - 2 * self.boxShift) * 3 / 4)
        )
    elif side == "opponent":
        self.playerSprite.nowHP -= damage
        if self.playerSprite.nowHP <= 0:
            self.playerSprite.nowHP = 0
            self.playerBool = False
        dmgMessage = pyText.TextWrap(
            (
                "You have dodged the attack from " + self.oppSprite.name + "!¥n¥n"
                if dodgeFlag else ""
            ) +
            (self.oppSprite.name + " has") + " dealt " +
            ((str(damage) + " damage") if damage else "no damage") + " on " +
            "You!" +
            (
                ("¥n¥n" + "You have died!")
                if not self.playerBool else ""
            ),
            self.BattleFont, int((self.width - 2 * self.boxShift) * 3 / 4)
        )
    else:
        print("Error occurred in DealDamage!")

    # Display the attack damage
    dmgSurf = pyText.TextRender(dmgMessage, self.BattleFont, self.boxSpacing, "center")
    dmgRect = dmgSurf.get_rect()
    dmgRect.center = (int(self.width / 2), self.displayHeight + int(self.boxHeight / 2))

    self.RefreshBattle()

    # Displays
    confirmed = False
    while not confirmed:

        if self.LoopBundle():
            break

        self.DisplayTextBox()

        self.screen.blit(dmgSurf, dmgRect)
        self.DisplayFloater()

        LoopBundle()


        for event in pygame.event.get():
            Quit(event)
            if event.type == KEYDOWN:
                if event.key == K_RETURN:
                    confirmed = True

# Displays a dialogue
def Dialogue(self, speaker = None, text = ""):

    nameSurf = self.BoldBattleFont.render(speaker if speaker else self.oppSprite.name, 1, color.
blue)

    displaySurf = pyText.TextWrap(text, self.BattleFont, self.width - 2 * self.boxShift)
    displaySurf = pyText.TextRender(displaySurf, self.BattleFont, int(self.boxSpacing * 3 / 4))
    displayRect = displaySurf.get_rect()
    displayRect.left = self.boxShift
    displayRect.bottom = int(self.height - (self.boxHeight - self.boxSpacing - displaySurf.get_
height() - self.textSize) / 2)

    confirmed = False
    while not confirmed:

        self.DisplayTextBox()
        self.screen.blit(nameSurf, (self.boxShift, displayRect.top - self.boxSpacing - self.tex
```

```
tSize))
            self.screen.blit(displaySurf, displayRect)
            self.DisplayFloater()

            for event in pygame.event.get():
                Quit(event)
                if event.type == KEYDOWN:
                    if event.key == K_RETURN:
                        confirmed = True

            LoopBundle()

    # Asks if the player would like to escape
    def PromptEscape(self):

        # Create Escape Menu
        escapePromptSurf = self.BoldBattleFont.render("Are you sure?", 1, color.blue)

        promptTopCoord = self.displayHeight + int(self.boxHeight / 2) - int(
            3 * (self.textSize + self.boxSpacing) / 2)
        escapeMenu = pyMenu.SelectionMenu(MenuArrow, self.selectionSurface, 0, self.boxSpacing)
        tempTopCoord = promptTopCoord + self.textSize + self.boxSpacing
        escapeMenu.CreateItem("No", self.BattleFont, (self.boxShift, tempTopCoord))
        tempTopCoord += self.textSize + self.boxSpacing
        escapeMenu.CreateItem("Yes", self.BattleFont, (self.boxShift, tempTopCoord))

        # Main Loop
        while True:

            if self.LoopBundle():
                break

            self.DisplayTextBox()
            self.screen.blit(escapePromptSurf, (self.boxShift, promptTopCoord))

            for event in pygame.event.get():
                Quit(event)

                result = escapeMenu.Scrolling(event)

                if result == 0:
                    return False
                elif result == 1:
                    return True

            escapeMenu.DisplayItem(self.screen)
            self.LoopBundle()
            LoopBundle()


class AdventureBattle(Battle):

    def __init__(self, screen, diff, playerInfo, bossInfo, BattleGraphics = None,
                 playerGraphics = None, initHP = -1, round = -1, LightName = False):

        self.playerInfo = playerInfo
        self.bossInfo = bossInfo.BossInfo()
        self.diff = diff

        if playerGraphics == None:
            self.playerGraphics = pygame.image.load(ProfileImage)
        else:
            self.playerGraphics = playerGraphics

        super().__init__(screen,
                         diff,
                         Sprite("You", self.playerInfo.fullHP(), self.playerGraphics, nowHP = initH
P),
                         self.bossInfo["sprite"], displayWallpaper=BattleGraphics, LightName = LightN
ame)
```

```
        self.round = round

    def StartDialogue(self):

        dialogue = self.bossInfo["dialogue"]
        for line in dialogue:
            self.Dialogue(line[0], line[1])

    def EndDialogue(self):

        dialogue = self.bossInfo["endDialogue"]
        for line in dialogue:
            self.Dialogue(line[0], line[1])

    def Reinitialize(self, bossInfo, diff = -1, round = -1, initHP = -1):
        self.playerBool = True
        self.oppBool = True

        self.bossInfo = bossInfo.BossInfo()
        self.oppSprite = self.bossInfo["sprite"]
        if initHP != -1:
            self.playerSprite.nowHP = initHP
        if diff != -1:
            self.diff = diff
        self.round = round

    def Main(self):

        self.RefreshBattle()
        # Pre-question
        escapeFlag, strengthFlag, defenseFlag = self.PreSelection()
        if escapeFlag:
            return "escape"
        else:
            pass

        # Generate a question
        questionObj = mQuestions.QuestionGenerator(self.diff, QuestionDatabase)
        questionObj = mDebug.EasyMode(questionObj)
        self.TimeDilation(questionObj)

        # Ask the question
        resultDict = self.DisplayQuestion(questionObj)

        # Calculate player AP
        bossDamage = self.CalculatePlayerAP(resultDict = resultDict,
                                            timeLimit = questionObj["time"],
                                            diffPt = questionObj["diffPt"],
                                            strengthFlag = strengthFlag)

        self.DealDamage("player", bossDamage)

        if not self.oppBool:
            return "win"
        else:
            pass

        # Calculate boss AP
        playerDamage = self.CalculateBossAP(diffPt = questionObj["diffPt"],
                                            defenseFlag = defenseFlag)
        self.DealDamage("opponent", playerDamage["damage"], playerDamage["dodgeFlag"])

        if not self.playerBool:
            return "lose"
        else:
            return "draw"

    # Before the player answers the questions, he can
    # choose to escape or drink a potion, dubbed PreSelection
    def PreSelection(self):
```

```
        promptSurf = self.BoldBattleFont.render("Choose an action:", 1, color.blue)

        # Create Menu
        promptTopCoord = self.displayHeight + int(self.boxHeight / 2) - int(4 * (self.textSize + sel
f.boxSpacing) / 2)
        mainMenu = pyMenu.SelectionMenu(MenuArrow, self.selectionSurface, 0, self.boxSpacing)
        tempTopCoord = promptTopCoord + self.textSize + self.boxSpacing
        mainMenu.CreateItem("Escape", self.BattleFont, (self.boxShift, tempTopCoord)) # 0
        tempTopCoord += self.textSize + self.boxSpacing
        mainMenu.CreateItem("Potion", self.BattleFont, (self.boxShift, tempTopCoord)) # 1
        tempTopCoord += self.textSize + self.boxSpacing
        mainMenu.CreateItem("Proceed to Question", self.BattleFont, (self.boxShift, tempTopCoord))
# 2

        # Create Potion Warning
        potionWarningDisplayRect = self.boxShift + self.BattleFont.size("Potion")[0] + self.textShi
ft + self.selectionHeight + self.boxSpacing¥
            , promptTopCoord + 2*(self.textSize + self.boxSpacing)
        potionWarning = self.BattleFont.render("You have already used a potion!", 1, color.red)

        potionUsed = False
        strengthFlag = False
        defenseFlag = False

        # mainMenu.SetCounter(2)

        # Main Loop
        while True:

            self.DisplayTextBox()
            self.screen.blit(promptSurf, (self.boxShift, promptTopCoord))

            for event in pygame.event.get():
                Quit(event)

                result = mainMenu.Scrolling(event)
                if result == 0:
                    if self.PromptEscape():
                        return True, 0, 0
                    else:
                        pass
                elif result == 1:
                    if potionUsed:
                        pass
                    else:
                        potionUsed, strengthFlag, defenseFlag = self.PromptPotionChoose()
                elif result == 2:
                    return False, strengthFlag, defenseFlag

            if potionUsed:

                self.screen.blit(potionWarning, potionWarningDisplayRect)

            mainMenu.DisplayItem(self.screen)
            LoopBundle()


    # Asks if the player would like to use a potion
    def PromptPotionChoose(self):

        # Create Potion Menu
        potionPromptSurf = self.BoldBattleFont.render("Choose a potion?", 1, color.blue)

        promptTopCoord = self.displayHeight + int(self.boxHeight / 2) - int(5 * (self.textSize + sel
f.boxSpacing) / 2)
        potionInfo = self.playerInfo.Potions({})
        potionMenu = pyMenu.SelectionMenu(MenuArrow, self.selectionSurface, 0, self.boxSpacing)
        tempTopCoord = promptTopCoord + self.textSize + self.boxSpacing
        potionMenu.CreateItem("[Qnty: " + str(potionInfo["regenPot"]) + "] Potion of Regeneration",
                              self.BattleFont, (self.boxShift, tempTopCoord))
        tempTopCoord += self.boxSpacing + self.textSize
```

```
        potionMenu.CreateItem("[Qnty: " + str(potionInfo["attackPot"]) + "] Potion of Braveness",
                              self.BattleFont, (self.boxShift, tempTopCoord))
        tempTopCoord += self.boxSpacing + self.textSize
        potionMenu.CreateItem("[Qnty: " + str(potionInfo["defendPot"]) + "] Potion of Perseverance
",
                              self.BattleFont, (self.boxShift, tempTopCoord))
        tempTopCoord += self.boxSpacing + self.textSize
        potionMenu.CreateItem("Exit",
                              self.BattleFont, (self.boxShift, tempTopCoord))

        # Main Loop
        while True:

            self.DisplayTextBox()

            for event in pygame.event.get():
                Quit(event)

                self.screen.blit(potionPromptSurf, (self.boxShift, promptTopCoord))

                result = potionMenu.Scrolling(event)
                if result == 0:
                    if potionInfo["regenPot"] > 0:
                        self.playerInfo.Potions({"regenPot" : -1})
                        self.playerSprite.nowHP += int(HealthPotionPerc * self.playerSprite.fullHP)
                        if self.playerSprite.nowHP > self.playerSprite.fullHP:
                            self.playerSprite.nowHP = self.playerSprite.fullHP
                        self.RefreshBattle()
                        return True, False, False
                elif result == 1:
                    if potionInfo["attackPot"] > 0:
                        self.playerInfo.Potions({"attackPot" : -1})
                        return True, True, False
                elif result == 2:
                    if potionInfo["defendPot"] > 0:
                        self.playerInfo.Potions({"defendPot" : -1})
                        return True, False, True
                elif result == 3:
                    return False, False, False

                potionMenu.DisplayItem(self.screen)
                LoopBundle()

    # Calculates the extra time allowed
    def TimeDilation(self, questionObj):

        questionObj["time"] = (self.playerInfo.timePerc() + 1) * questionObj["time"]

    # Calculate how much damage is done to the boss
    def CalculatePlayerAP(self, resultDict, timeLimit, diffPt, strengthFlag):
        if resultDict["result"] == False or resultDict["timeLeft"] == 0:
            return 0
        else:

            # Base damage
            dmg = self.playerInfo.fullAP()
            mDebug.Damage(dmg)

            # Random factor
            dmg *= (0.99 + random.random() * 0.02)
            mDebug.Damage(dmg)

            # Player Perks
            dmg *= 1 + self.playerInfo.swordPerc()
            mDebug.Damage(dmg)

            # Time Left
            dmg *= (1/3) * (resultDict["timeLeft"] / timeLimit + 2)
            mDebug.Damage(dmg)

            # Difficulty Variation
```

```
            dmg *= 1 + diffPt
            mDebug.Damage(dmg)

            # Strength Potion
            if strengthFlag:
                dmg *= 1 + AttackPotionPerc
            mDebug.Damage(dmg)

            return math.ceil(dmg)

    # Calculate how much damage is done to the player
    def CalculateBossAP(self, diffPt, defenseFlag):

        # Agility Perk
        if random.random() < self.playerInfo.dodgePerc():

            return {
                "damage" : 0,
                "dodgeFlag" : True
            }

        else:

            # Base damage
            dmg = self.bossInfo["fullAP"]
            mDebug.Damage(dmg)

            # Random factor
            dmg *= 0.99 + random.random() * 0.02
            mDebug.Damage(dmg)

            # Player Perks
            dmg *= 1 - self.playerInfo.chestPerc()
            mDebug.Damage(dmg)
            dmg *= 1 - self.playerInfo.legPerc()
            mDebug.Damage(dmg)
            dmg *= 1 - self.playerInfo.defensePerc()
            mDebug.Damage(dmg)

            # Difficulty Variation
            dmg *= 1 - diffPt
            mDebug.Damage(dmg)

            # Strength Potion
            if defenseFlag:
                dmg *= 1 - DefendPotionPerc


            return {
                "damage": math.ceil(dmg),
                "dodgeFlag": False
            }

    # Displays a dialogue
    def FightContinues(self):
        continueMessageSurf = self.BattleFont.render("The fight continues...", 1, color.blue)
        continueMessageRect = (int((- continueMessageSurf.get_width() + self.width) / 2),
                               self.displayHeight + int((- continueMessageSurf.get_height() + self.boxH
eight) / 2))
        confirmed = False
        while not confirmed:

            self.DisplayTextBox()
            self.screen.blit(continueMessageSurf, continueMessageRect)
            self.DisplayFloater()

            for event in pygame.event.get():
                Quit(event)
                if event.type == KEYDOWN:
                    if event.key == K_RETURN:
                        confirmed = True
```

```
            LoopBundle()

    # Displays a dialogue
    def BossAppears(self, bossType = "Civilian"):

        self.RefreshBattle()

        bossTypeSurf = self.BoldBattleFont.render(bossType, 1, color.black) if ¥
            bossType == "Final Boss" else self.BattleFont.render(bossType, 1, color.black)
        hasAppearedSurf = self.BoldBattleFont.render("has appeared!", 1, color.black) if ¥
            bossType == "Final Boss" else self.BattleFont.render("has appeared!", 1, color.black)
        bossNameSurf = pygame.font.Font(BattleFontString, self.textSize*2).¥
            render(self.bossInfo["sprite"].name,
                  1, color.red if bossType == "Final Boss" else color.blue)

        bossNameRect = bossNameSurf.get_rect()
        bossNameRect.center = (int(self.width / 2), self.displayHeight + int(self.boxHeight / 2))

        bossTypeRect = bossTypeSurf.get_rect()
        bossTypeRect.bottom = bossNameRect.top - self.boxSpacing
        bossTypeRect.centerx = int(self.width / 2)

        hasAppearedRect = hasAppearedSurf.get_rect()
        hasAppearedRect.top = bossNameRect.bottom + self.boxSpacing
        hasAppearedRect.centerx = int(self.width / 2)

        confirmed = False
        while not confirmed:

            self.DisplayTextBox()

            self.screen.blit(bossNameSurf, bossNameRect)
            self.screen.blit(bossTypeSurf, bossTypeRect)
            self.screen.blit(hasAppearedSurf, hasAppearedRect)

            self.DisplayFloater()


            for event in pygame.event.get():
                Quit(event)
                if event.type == KEYDOWN:
                    if event.key == K_RETURN:
                        confirmed = True

            LoopBundle()

    # Displays a dialogue (for Grinding)
    def PromptContinue(self):

        msg = "Would you like to continue grinding?"
        msgSurf = self.BattleFont.render(msg, 1, color.blue)

        promptTopCoord = self.displayHeight + int(self.boxHeight / 2) - int(3 * (self.textSize + sel
f.boxSpacing) / 2)
        escapeMenu = pyMenu.SelectionMenu(MenuArrow, self.selectionSurface, 0, self.boxSpacing)
        tempTopCoord = promptTopCoord + self.textSize + self.boxSpacing
        escapeMenu.CreateItem("Yes", self.BattleFont, (self.boxShift, tempTopCoord))
        tempTopCoord += self.textSize + self.boxSpacing
        escapeMenu.CreateItem("No", self.BattleFont, (self.boxShift, tempTopCoord))

        # Main Loop
        while True:

            self.DisplayTextBox()

            for event in pygame.event.get():
                Quit(event)

                self.screen.blit(msgSurf, (self.boxShift, promptTopCoord))
                result = escapeMenu.Scrolling(event)
```

```
            if result == 0:
                return True
            elif result == 1:
                return False

            escapeMenu.DisplayItem(self.screen)
            LoopBundle()

    # Displays rewards
    def Rewards(self, congratText = False):

        displayText = ""

        if congratText:
            displayText += "Congratulations on defeating ", self.oppSprite.name, ", Level ", str(sel
f.diff) + "!¥n"

        expGained = int(self.bossInfo["rewards"]["exp"] * (0.97 + 0.1 * random.random()))
        cashGained = int(self.bossInfo["rewards"]["cash"] * (0.97 + 0.1 * random.random()))
        displayText += "You have earned " + str(expGained) + ¥
                    " Experience Points and " + str(cashGained) + " Cash."

        expDict = self.playerInfo.Experience(expGained)
        self.playerInfo.Cash(cashGained)

        if expDict["levelUp"]:
            displayText += "¥nYou have levelled up to Level " + str(expDict["level"]) + "!"

        displaySurf = pyText.TextWrap(displayText, self.BattleFont, self.width - 2 * self.boxShift)
        displaySurf = pyText.TextRender(displaySurf, self.BattleFont, self.boxSpacing, alignment="c
enter")
        displayRect = displaySurf.get_rect()
        displayRect.center = (int(self.width / 2), int(self.displayHeight + self.boxHeight / 2))

        confirmed = False
        while not confirmed:

            self.DisplayTextBox()
            self.screen.blit(displaySurf, displayRect)
            self.DisplayFloater()

            for event in pygame.event.get():
                Quit(event)
                if event.type == KEYDOWN:
                    if event.key == K_RETURN:
                        confirmed = True

            LoopBundle()


class MultiplayerBattle(Battle):

    PlayerHP = PlayerHP
    PlayerAP = AttackDmg
    RandomRange = 0.05
    TimerRange = 1 / 3

    def __init__(self, screen, playerSprite = None, oppSprite = None, BattleGraphics = None, LightN
ame = False):

        if playerSprite:
            self.playerSprite = playerSprite
        else:
            self.playerSprite = Sprite("Player 1", self.PlayerHP, pygame.image.load(ProfileImage))

        if oppSprite:
            self.oppSprite = oppSprite
        else:
            self.oppSprite = Sprite("Player 2", self.PlayerHP, pygame.image.load(Player2Image))
```

```
        super().__init__(screen, 1, self.playerSprite, self.oppSprite,
                         displayWallpaper = BattleGraphics, LightName = LightName)

        self.turn = 1

    # Chooses difficulty and returns the difficulty
    def DifficultySelection(self):

        # Create Selection Menu
        diffPromptSurf = self.BoldBattleFont.render("Choose a difficulty.", 1, color.blue)

        promptTopCoord = self.displayHeight + int(self.boxHeight / 2) - int(
            5 * (self.textSize + self.boxSpacing) / 2)

        diffMenu = pyMenu.SelectionMenu(MenuArrow, self.selectionSurface, 0, self.boxSpacing)
        tempTopCoord = promptTopCoord + self.textSize + self.boxSpacing
        diffMenu.CreateItem("(Level 1) Quadratic Equations and Graphs",
                            self.BattleFont, (self.boxShift, tempTopCoord))
        tempTopCoord += self.boxSpacing + self.textSize
        diffMenu.CreateItem("(Level 2) Arithmetic Sequences",
                            self.BattleFont, (self.boxShift, tempTopCoord))
        tempTopCoord += self.boxSpacing + self.textSize
        diffMenu.CreateItem("(Level 3) Geometric Sequence and Polynomials",
                            self.BattleFont, (self.boxShift, tempTopCoord))
        tempTopCoord += self.boxSpacing + self.textSize
        diffMenu.CreateItem("Escape",
                            self.BattleFont, (self.boxShift, tempTopCoord))

        # Main Loop
        while True:

            if self.LoopBundle():
                break

            self.DisplayTextBox()
            self.screen.blit(diffPromptSurf, (self.boxShift, promptTopCoord))

            for event in pygame.event.get():
                Quit(event)

                result = diffMenu.Scrolling(event)
                if result == 0:

                    return 1
                elif result == 1:

                    return 2
                elif result == 2:

                    return 3
                elif result == 3:
                    if self.PromptEscape():
                        return -1

            diffMenu.DisplayItem(self.screen)
            LoopBundle()

    def CalculateAP(self, resultDict, timeLimit, diffPt):

        result = resultDict["result"]
        timeLeft = resultDict["timeLeft"]
        if result and timeLeft:

            dmg = self.PlayerAP[self.diff]

            dmg *= 1 - self.RandomRange / 2 + self.RandomRange * random.random()

            dmg *= (1 - self.TimerRange) + self.TimerRange * timeLeft / timeLimit

            dmg *= 1 + diffPt
```

```
            return int(dmg)

        else:
            return 0

    def ChangeRound(self):
        self.turn = 1 if self.turn == 2 else 2


class OfflineBattle(MultiplayerBattle):

    def __init__(self, screen, playerSprite = None, oppSprite = None):

        super().__init__(screen, playerSprite, oppSprite, DefaultMultiOffline)


    def Main(self):

        self.RefreshBattle()

        self.Dialogue("Narrator", "A quarrel between friends does not make them enemies; it's a type
 of communication. "
                                  + "Go, my dear, and demonstrate your intellect.")

        roundCounter = 0

        run = True
        while run:

            if self.playerBool and self.oppBool:
                if roundCounter:
                    self.ChangeRound()
            else:
                run = False
                winner = self.turn
                break

            self.Dialogue("Narrator", "It is now your turn, " + (self.playerSprite.name if self.turn
 == 1 else self.oppSprite.name))
            self.diff = self.DifficultySelection()
            if self.diff == -1:
                self.Dialogue("Narrator", (self.playerSprite.name if self.turn == 1 else self.oppSpr
ite.name) + " "
                              + "has abandoned.")
                run = False
                return
            question = mQuestions.QuestionGenerator(self.diff, QuestionDatabase)
            question = mDebug.EasyMode(question)
            resultDict = self.DisplayQuestion(question)
            dmg = self.CalculateAP(resultDict, question["time"], question["diffPt"])
            self.DealDamage("player" if self.turn == 1 else "opponent", dmg)
            roundCounter += 1

        self.Dialogue("Narrator", "Well done, " + (self.playerSprite.name if self.turn == 1 else sel
f.oppSprite.name) + "!¥n"
                      + "Now, don't brag just because you won, and don't cry because you lost. It takes
 time "
                      + "to become a great Mathematician, and I believe both of you can do it!")


# ************************************************************************************************
**************
# *********  The following are display modules used in the above classes.  ************************
************
# ************************************************************************************************
**************


def BattleDisplay(playerSprite, oppSprite, dim, round = -1, playerBool = True, oppBool = True, Name
TagBg = None, LightName = False):
```

```
    if LightName:
        color = (255, 249, 224)
    else:
        color = (26, 14, 0)

    if NameTagBg == None:
        NameTagWallpaper = DefaultBattleNameTag
    else:
        NameTagWallpaper = NameTagBg

    # Defining the surface that is going to be returned
    Surf = pygame.Surface(dim, pygame.SRCALPHA)
    height = dim[1]
    width = dim[0]

    # Health bar creation
    # Height : 55
    barHeight = int((55/350) * height)
    barWidth = int((400/900) * width)
    playerBar = HealthBar(playerSprite.fullHP, playerSprite.nowHP, (barWidth, barHeight), "left")
    if oppSprite.fullHP == 0:
        oppBar = pygame.Surface((1, 1), pygame.SRCALPHA)
    else:
        oppBar = HealthBar(oppSprite.fullHP, oppSprite.nowHP, (barWidth, barHeight), "right")

    # Text
    # Height = 31 + 5 * 2 divided by 2
    textHeight = int((30/350) * height)
    playerTag = BattleFont.render(playerSprite.name, 1, color)
    oppTag = BattleFont.render(oppSprite.name, 1, color)
    playerTag = ScaleGraphics(playerTag, textHeight, False, True)
    oppTag = ScaleGraphics(oppTag, textHeight, False, True)
    playerSize = playerTag.get_size()
    oppSize = oppTag.get_size()
    boxVertShift = int((7.5/350) * height)
    boxHorzShift = int((4/900) * width)
    playerName = pygame.transform.scale(NameTagWallpaper, (playerSize[0] + 2 * boxHorzShift, playe
rSize[1] + 2 * boxVertShift))
    oppName = pygame.transform.scale(NameTagWallpaper, (oppSize[0] + 2 * boxHorzShift, oppSize[1] +
 2 * boxVertShift))
    playerName.blit(playerTag, (boxHorzShift, boxVertShift))
    oppName.blit(oppTag, (boxHorzShift, boxVertShift))

    if round > 0:
        roundTag = BattleFont.render("Round: " + str(round) + "/2", 1, color)
        roundTag = ScaleGraphics(roundTag, textHeight, False, True)
        roundSize = roundTag.get_size()
        roundName = pygame.transform.scale(NameTagWallpaper, (roundSize[0] + 2 * boxHorzShift, roun
dSize[1] + 2 * boxVertShift))
        roundName.blit(roundTag, (boxHorzShift, boxVertShift))
        roundName = ScaleGraphics(roundName, textHeight, False, True)

    # Blitting

    # Health Bars
    # Height : 15
    topCoord = int((15/350) * height)
    HorzShift = int((30/900) * width)
    Surf.blit(playerBar, (HorzShift, topCoord))
    Surf.blit(oppBar, (width - HorzShift - barWidth, topCoord))

    # Name Tag
    # Height : 5
    topCoord += int((5/350) * height) + barHeight
    nameCoord = topCoord

    # Graphics
    topCoord += int((5/350) * height) + int((textHeight + 2 * boxVertShift) / 2)
    graphicsHeight = height - topCoord
    playerGraphics = playerSprite.returnSurface(graphicsHeight)
    oppGraphics = oppSprite.returnSurface(graphicsHeight)
```

```
    graphicsShift = HorzShift + int((350/900/2) * width)
    playerRect = playerGraphics.get_rect()
    playerRect.center = (graphicsShift, topCoord + int(graphicsHeight // 2))
    oppRect = oppGraphics.get_rect()
    oppRect.center = (width - graphicsShift, topCoord + int(graphicsHeight // 2))
    if playerBool:
        Surf.blit(playerGraphics, playerRect)
    if oppBool:
        Surf.blit(oppGraphics, oppRect)

    # Name Tag
    Surf.blit(playerName, (HorzShift, nameCoord))
    if oppSprite.name:
        Surf.blit(oppName, (width - HorzShift - oppName.get_width(), nameCoord))
    if round > 0:
        tempRect = roundName.get_rect()
        tempRect.centerx = int(width // 2)
        tempRect.top = nameCoord
        Surf.blit(roundName, tempRect)

    return Surf

def HealthBar(fullHP, playerHP, dim, side): # --> Surface

    height = dim[1]
    width = dim[0]

    # Define a surface and fill in default color/image
    drawSurface = pygame.Surface(dim, pygame.SRCALPHA)

    # Heart blit at (0,0)
    # Health bars blit at height / 2, 0
    # Health bar width is width - height / 2
    # Text blits at height
    heartWidth = int(height // 2)
    barWidth = width - int(height//2)
    barHeight = int(height * 0.8)
    textHeight = int(barHeight * 11 / 13 * 0.65)

    # Correct Dimensions
    drawHeart = pygame.transform.scale(Heart, (height, height))
    drawHealthBg = pygame.transform.scale(HealthBarBackdrop, (barWidth, barHeight))
    drawHealthBar = pygame.transform.scale(HealthBarOverlay, (barWidth, barHeight))
    if side == "right":
        drawHealthBg = pygame.transform.flip(drawHealthBg, True, False)
        drawHealthBar = pygame.transform.flip(drawHealthBar, True, False)

    # Calculate dimensions
    ratio = playerHP / fullHP
    borderWidth = barWidth / 100
    healthWidth = ratio * (barWidth - 2 * borderWidth)
    if side == "left":
        healthStart = (0, 0)
        heartDim = (int(borderWidth + healthWidth), barHeight)
        newBar = Crop(drawHealthBar, healthStart, heartDim)
    else:
        healthStart = (int(barWidth - borderWidth - healthWidth), 0)
        heartDim = (int(borderWidth + healthWidth), barHeight)
        newBar = Crop(drawHealthBar, healthStart, heartDim)


    # Font and text
    text = "Health: " + str(playerHP) + "/" + str(fullHP)
    textSurf = BattleFont.render(text, 1, (35, 16, 5))
    conversionRatio = textHeight/FONTHEIGHT
    textSurf = pygame.transform.scale(textSurf, (int(textSurf.get_width() * conversionRatio), text
Height))

    # Blit the bar on the output surface
    if side == "left":
        drawSurface.blit(drawHealthBg, (heartWidth, int((height - barHeight) // 2)))
```

```
        drawSurface.blit(newBar, (heartWidth, int((height - barHeight) // 2)))
        drawSurface.blit(drawHeart, (0, 0))
        drawSurface.blit(textSurf, (int((height) // 2 + barWidth - textSurf.get_width() - 4 * border
Width), int((height - textSurf.get_height()) // 2)))
    else:
        drawSurface.blit(drawHealthBg, (0, int((height - barHeight) // 2)))
        drawSurface.blit(newBar, (int(barWidth - healthWidth - borderWidth), int((height - barHeigh
t) // 2)))
        drawSurface.blit(drawHeart, (width - height, 0))
        drawSurface.blit(textSurf, (int(4 * borderWidth), int((height - textSurf.get_height()) //
2)))

    return drawSurface



# Note that TestPlayer.txt is used for
# testing player data
def main2():

    import APlayer
    import ABoss

    PlayerInfo = APlayer.LoadSave("TestPlayer.txt")
    BossList = ABoss.ReadBoss("Bosses.txt")
    BossInfo = BossList[1][1]

    screen = pygame.display.set_mode((1280, 800))

    bossBattle = AdventureBattle(screen, 1, PlayerInfo, BossInfo, initHP = 1000)

    bossBattle.Main()

def main1():
    pygame.init()
    pygame.font.init()

    # Hikari = pygame.transform.flip(pygame.image.load("assets/Hikari2.png"), True, False)
    Hikari = pygame.transform.flip(pygame.image.load("assets/JPEG/Hikari2.jpg"), True, False)
    # Tairitsu = pygame.transform.flip(pygame.image.load("assets/Tairitsu2.png"), True, False)
    Tairitsu = pygame.transform.flip(pygame.image.load("assets/JPEG/Tairitsu2.jpg"), True, False)

    kitten = Hikari
    dog = Tairitsu

    playerSprite = Sprite("Kitten", 1000, kitten)
    oppSprite = Sprite("Dog", 10000, dog)

    screen = pygame.display.set_mode((1280, 680))

    '''
    battle = BattleDisplay(playerSprite, oppSprite, (900, 250))
    BattleWallpaperTemp = pygame.transform.scale(BattleWallpaper,(900, 350))
    BattleBoxTemp = pygame.transform.scale(BattleBox, (900, 250))
    '''

    # somethingevennewer = HealthBar(1000, 500, (500, 100), "right")
    # screen.blit(somethingevennewer, (0,0))

    battle = Battle(screen, 1, playerSprite, oppSprite)
    battle.Main()

    k = 1

    while True:

        LoopBundle()

        '''
        screen.blit(BattleBoxTemp, (0, 350))
        screen.blit(BattleWallpaperTemp, (0, 0))
```

```
        screen.blit(battle, (0, 0))

        LoopBundle()

        playerSprite.nowHP %= 1000
        playerSprite.nowHP += 10
        oppSprite.nowHP %= 10000
        oppSprite.nowHP += 10
        battle = BattleDisplay(playerSprite, oppSprite, (900, 250), 1000)
        '''

        '''
        k %= 1000
        k += 10
        screen.fill((0,0,0))
        somethingevennewer = HealthBar(1000, k, (500, 100), "left")
        screen.blit(somethingevennewer, (0, 0))
        LoopBundle()
        for i in pygame.event.get():

            pass

        '''

if __name__ == "__main__":
    pass
```

10. ABoss.py

```
"""
ABoss.py
"""


import json
import pygame
import mSprite
import mErrors
import random

class Boss:

    __slots__ = "name", "fullHP", "graphics", "sprite", "fullAP", "rewards", "dialogue", "endDialog
ue"

    def __init__(self, name = "", fullHP = 0, fullAP = 0, graphics = "", rewards = {}, dialogue = [],
 endDialogue = []):

        self.name = name
        self.fullHP = fullHP
        self.graphics = graphics # str
        self.fullAP = fullAP
        self.rewards = rewards
        self.dialogue = dialogue
        self.endDialogue = endDialogue

    def BossInfo(self):

        tempDict = {}
        tempDict["sprite"] = mSprite.Sprite(self.name, self.fullHP, pygame.image.load(self.graphic
s)
        if self.graphics else pygame.Surface((1, 1), pygame.SRCALPHA))
        tempDict["fullAP"] = self.fullAP
        tempDict["rewards"] = self.rewards
        tempDict["dialogue"] = self.dialogue
        tempDict["endDialogue"] = self.endDialogue

        return tempDict

def WriteBoss(filename):

    try:
        bossFile = open(filename, 'r')
        bossSomething = json.loads(bossFile.read())
        bossList = bossSomething["Bosses"]
    except:
        bossList = []

    run = True
    while run:

        print("Please input a boss: (Insert name = -1 to end, leave Graphics as blank if no image)")
        diff = int(input(f"{'Difficulty (diff):':30} "))
        if diff == -1:
            run = False
            break
        name = input(f"{'Name (name):':30} ")
        fullHP = int(input(f"{'Full Health (fullHP)':30} "))
        fullAP = int(input(f"{'Full Attack (fullAP)':30} "))
        graphics = input(f"{'Graphics File Path (graphics)':30} ")
        print("Rewards: ")
        exp = int(input(f"{'Experience (exp)':30} "))
        cash = int(input(f"{'Cash (cash)':30} "))

        dialogue = []
        run = True
        while run:
            print("Dialogue (dialogue):")
```

```
            speaker = input(f"{'Speaker':30} ")
            if speaker == "-1":
                run = False
                break
            speech = input(f"{'Speech':30} ")
            dialogue.append([speaker, speech])

        endDialogue = []
        run = True
        while run:
            print("End Dialogue (endDialogue):")
            speaker = input(f"{'Speaker':30} ")
            if speaker == "-1":
                run = False
                break
            speech = input(f"{'Speech':30} ")
            endDialogue.append([speaker, speech])


        bossList.append({
            "diff": diff,
            "name": name,
            "fullHP": fullHP,
            "fullAP": fullAP,
            "graphics": graphics,
            "rewards": {
                "exp" : exp,
                "cash": cash
            },
            "dialogue": dialogue,
            "endDialogue": endDialogue
        })

    bossSomething = {"Bosses" : bossList}
    storeStr = json.dumps(bossSomething, indent = 4)

    bossFile = open(filename, 'w')
    bossFile.write(storeStr)

def ReadBoss(filename):

    try:
        bossFile = open(filename, 'r')
        bossJson = json.loads(bossFile.read())
        bossList = bossJson["Bosses"]
    except:
        raise mErrors.BossError
    else:
        BossDict = {}
        for i in range(1, 8):
            BossDict[i] = []
        for i in bossList:
            BossDict[i["diff"]].append(
                # !!!!勝手に何をやっているのだよお前は!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                Boss(i["name"], int(i["fullHP"]*(0.9 if i["name"] != "yandere" else 1)), i["fullAP"],
 #!!!!!!!!!勝手に!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                     #!!!!!!!!!!!!!!!!!!!変えるなよ!!!!!!!!!!!!!!!!おおおおおおおおおおおお
い!!!!!!!!!!!!!!!!
                     i["graphics"],
                     i["rewards"], i["dialogue"], i["endDialogue"])
            )

        return BossDict

def BossFetch(diff, bossDict, index = -1):
    if index != -1:
        return bossDict[diff][int(index)]
    else:
        boss = bossDict[diff][random.randint(0, len(bossDict[diff]) - 1)]
        if boss.name == "yandere":
```

```
            if random.random() > 0.10:
                return BossFetch(diff, bossDict, index)
            else:
                return boss
        else:
            return boss


def main():

    # WriteBoss("Bosses.txt")
    ReadBoss("files/Bosses.txt")

if __name__ == "__main__":


    main()
```

11. ABossBattle.py

```
"""
ABossBattle.py
"""

import Main_Battle as Battle
import pygame
from GlobalVar import *
import pyAnimation
import ATextBox
import ASequence

def a_DoBoss(screen, playerInfo):

    # Difficulty
    diff = playerInfo.Beated(False)["bosslevel"]

    # Counter for the boss
    counter = -1
    battleResults = "win"

    if diff == 3:
        LightName = True
    else:
        LightName = False

    # Initiate bossInfo, Pass in boss information
    bossInfo = ABoss.Boss()
    BattleClass = Battle.AdventureBattle(screen, diff,
                                         playerInfo, bossInfo, BattleGraphics=DefaultBattleWallpaper[dif
f],
                                         LightName = LightName)

    pyAnimation.EnterFade(screen, BattleClass.Fade, CentralClock, GameFPS)

    BattleClass.Dialogue("Narrator", "You travelled through the world to beat Mathematicians¥n" +
                         "Here and now, you will face the man of your dreams.")

    # Loop for incrementing the boss
    while counter <= 2:

        if battleResults == "escape":
            return
        elif battleResults == "lose":
            return
        elif battleResults == "draw":
            BattleClass.FightContinues()
        elif battleResults == "win":
            if counter != -1:
                BattleClass.EndDialogue()
            if counter + 1 <= 1:
                # Increment counter
                counter += 1

                # Pass in new boss info
                bossInfo = BossDatabase[diff][counter]
                BattleClass.Reinitialize(bossInfo, diff = diff + (3 if counter == 1 else 0), round =
counter + 1)

                # Display boss
                BattleClass.BossAppears("Final Boss" if counter == 1 else "Mathematician")
                BattleClass.StartDialogue()
            else:
                break

        # Battle sequence
        battleResults = BattleClass.Main()
```

```
    # Reads player info
    playerInfoDict = playerInfo.Beated(0)

    # Rewards will not be given if the player has beaten the boss once before
    if playerInfoDict["beat"]:

        # Asks whether the player would like to watch the epilogue again
        popUp = ATextBox.PopUp(screen)
        popUp.TextBox.InsertText("Would you like to watch the Epilogue again?", "center", True)
        nah = popUp.TextBox.InsertMenu("No")
        ya = popUp.TextBox.InsertMenu("Yes")
        popUp.TextBox.Finalize()

        popUp.Start()

        run = True
        while run:

            for event in pygame.event.get():
                Quit(event)
                result = popUp.TextBox.Event(event, popUp.Offset())
                if result == nah:
                    epilogue = False
                    run = False
                elif result == ya:
                    epilogue = True
                    run = False

            popUp.Blit()
            LoopBundle()
    # If the player hasn't beat the boss before
    else:
        BattleClass.Rewards()
        if playerInfoDict["bosslevel"] == 3:
            epilogue = True
        else:
            epilogue = False

    playerInfo.Beated(1)

    # The player has defeated all three bosses
    if epilogue:
        playerInfo.BeatFlag()
        return ASequence.Epilogue(screen)
    else:
        return False

def main():
    import APlayer
    screen = pygame.display.set_mode((1280, 800))
    playerInfo = APlayer.LoadSave("TestPlayer.txt")
    a_DoBoss(screen, playerInfo)

if __name__ == "__main__":
    main()
```

12. AGrinding.py

```
"""
AGrinding.py
"""

import Main_Battle as Battle
import pygame
import random
import ABoss
from GlobalVar import *
import pyAnimation

def a_DoGrinding(screen, playerInfo):

    # Difficulty
    diff = playerInfo.Beated(False)["bosslevel"]

    # Counter for the boss
    BattleClass = Battle.AdventureBattle(screen, diff, playerInfo, ABoss.Boss(),
                                         BattleGraphics=DefaultGrindWallpaper)

    pyAnimation.EnterFade(screen, BattleClass.Fade, CentralClock, GameFPS)

    BattleClass.Dialogue("Narrator", "'Practice makes perfect.'¥n" +
                                     "You mumbled these words as you bravely approached a stranger, pick
ing them for a fight.")


    # Initiate bossInfo, Pass in boss information
    playerInfoDict = playerInfo.Beated(False)
    if playerInfoDict["beat"]:
        randomDiff = 7
    else:
        randomDiff = diff+3
    bossInfo = ABoss.BossFetch(randomDiff, BossDatabase)
    BattleClass.Reinitialize(bossInfo, randomDiff)

    # Display boss
    BattleClass.BossAppears()
    BattleClass.StartDialogue()

    # Loop for incrementing the boss
    cont = True
    while cont:

        # Battle sequence
        battleResults = BattleClass.Main()

        if battleResults == "escape":
            return
        elif battleResults == "lose":
            return
        elif battleResults == "draw":
            BattleClass.FightContinues()
        elif battleResults == "win":
            # Rewards of previous boss
            BattleClass.EndDialogue()
            BattleClass.Rewards()

            cont = BattleClass.PromptContinue()
            if cont:

                # Pass in new boss info
                if playerInfoDict["beat"]:
                    randomDiff = 7
                else:
                    randomDiff = diff + 3
                print(randomDiff)
                bossInfo = ABoss.BossFetch(randomDiff, BossDatabase)
```

```
            BattleClass.Dialogue("Narrator", "You took a short rest before facing your next oppon
ent.")

            BattleClass.Reinitialize(bossInfo, randomDiff, initHP=playerInfo.fullHP())

            # Display boss
            BattleClass.BossAppears()
            BattleClass.StartDialogue()
        else:
            break

def main():
    import pygame
    import APlayer
    screen = pygame.display.set_mode((1280, 800))
    playerInfo = APlayer.LoadSave("TestPlayer.txt")
    a_DoGrinding(screen, playerInfo)

if __name__ == "__main__":
    main()
```

13. ALoad.py

```
"""
ALoad.py
"""


import pyMenu
from GlobalVar import *
import APlayer
import pyAnimation

def LoadOptions(screen):

    menu = pyMenu.FrameMenu(screen)

    menu.AddGraphics(pygame.image.load(Floppy))

    menu.InsertText("Load Options", Title = True)
    menu.InsertText("Please choose a load option.", Text=True)

    menu.InitiateMenu()
    one = menu.InsertMenu("New Game")
    menu.InsertText("Play the game from the start", Explanation=True)
    menu.InsertText("", Enter=True)
    two = menu.InsertMenu("Load from Save")
    menu.InsertText("Play from where you last left off", Explanation=True)
    menu.InsertText("", Enter=True)
    three = menu.InsertMenu("Return to Main Menu")

    displayFunctionSurf = menu.DisplayFunctionSurf()
    def display2Function():
        screen.blit(displayFunctionSurf, (0, 0))

    def displayFunction():
        screen.fill(color.white)
        menu.Blit()

    pyAnimation.EnterFade(screen, display2Function, CentralClock, GameFPS)

    while True:

        displayFunction()

        for event in pygame.event.get():
               Quit(event)
               counter = menu.Scrolling(event)
               if counter == one:
                   return APlayer.Initialize(), True, True
               elif counter == two:
                   return APlayer.LoadSave(PlayerDataFileName), False, True
               elif counter == three:
                   return None, None, False
        LoopBundle()
```

14. AMenu.py

```
"""
AMenu.py
"""

import APlayer
import pygame
import pyMenu
import AShop
from pygame.locals import *
from GlobalVar import *
import APlayer
import pyImage
import pyAnimation

def a_Menu(screen, playerInfo):

    menu = pyMenu.FrameMenu(screen)

    menu.AddGraphics(pygame.image.load(Settings))

    menu.InsertText("Game Menu", Title = True)

    menu.InitiateMenu()
    menu.InsertMenu("View Statistics")
    menu.InsertMenu("Save Game")
    menu.InsertMenu("Quit Game")

    menu.InsertText("", Enter=True)
    menu.InsertMenu("Return to Village")

    def displayFunction():
        screen.fill(color.white)
        menu.Blit()

    displayFunctionSurf = menu.DisplayFunctionSurf()
    def display2Function():
        screen.blit(displayFunctionSurf, (0, 0))

    # pyAnimation.EnterFade(screen, display2Function, CentralClock, GameFPS)

    while True:

        displayFunction()

        for event in pygame.event.get():
               Quit(event)
               counter = menu.Scrolling(event)
               if counter == 0:
                   # View stats
                   a_Stats(screen, playerInfo)
               elif counter == 1:
                   # Save game
                   a_Save(screen, playerInfo)
               elif counter == 2:
                   # Quit game
                   exitFlag = a_QuitMenu(screen)
                   if exitFlag:
                       return True
               elif counter == 3:
                   return False

        LoopBundle()

def a_QuitMenu(screen):

    menu = pyMenu.FrameMenu(screen)

    menu.InsertText("Quit Game", Title = True)
    menu.InsertText("Are you sure you want to quit the game?", Header = True)
```

```
    menu.InsertText("Unsaved data will be discarded.", Explanation = True)
    menu.InsertText("", Enter=True)

    menu.InitiateMenu()
    menu.InsertMenu("No")
    menu.InsertMenu("Yes")

    while True:
        screen.fill(color.white)
        menu.Blit()
        for event in pygame.event.get():
            Quit(event)
            counter = menu.Scrolling(event)
            if counter == 0:
                return False
            elif counter == 1:
                return True
        LoopBundle()

def a_Save(screen, playerInfo):

    def Exists(filename):

        try:
            file = open(filename, 'r')
        except:
            return False
        else:
            return True

    def Confirmation(screen):

        subMenu = pyMenu.FrameMenu(screen)

        subMenu.AddGraphics(pygame.image.load(Floppy))

        subMenu.InsertText("Save Game", Title=True)
        subMenu.InsertText("The old save is not empty.", Text=True)
        subMenu.InsertText("Would you like to overwrite the old save?", Text=True)
        subMenu.InsertText("", Enter=True)

        subMenu.InitiateMenu()
        subMenu.InsertMenu("No")
        subMenu.InsertMenu("Yes")

        while True:
            screen.fill(color.white)
            subMenu.Blit()
            for event in pygame.event.get():
                Quit(event)
                counter = subMenu.Scrolling(event)
                if counter == 0:
                    return False
                elif counter == 1:
                    return True
            LoopBundle()

    menu = pyMenu.FrameMenu(screen)
    menu.AddGraphics(pygame.image.load(Floppy))
    filename = PlayerDataFileName

    if Exists(filename):
        check = Confirmation(screen)
        if check:
            pass
        else:
            return
    else:
        pass

    # Saving procedure
```

```
    APlayer.SaveFile(filename, playerInfo)

    menu.InsertText("Save Game", Title=True)
    menu.InsertText("", Enter=True)
    menu.InsertText("", Enter=True)
    menu.TextWrap("Game data has been saved successfully.", Header=True)
    menu.InsertText("You may safely quit the game.", Header=True)
    menu.InsertText("(Press Enter to continue)", Explanation=True)

    while True:
        screen.fill(color.white)
        menu.Blit()
        for event in pygame.event.get():
            Quit(event)
            if event.type == KEYDOWN:
                if event.key == K_RETURN:
                    return
        LoopBundle()

def a_Stats(screen, playerInfo):

    expDict = playerInfo.Experience()
    playerSkill = playerInfo.Skills()
    equipmentDict = playerInfo.Equipment()
    potionDict = playerInfo.Potions()

    def Stat(pageNumber):

        screen.fill(color.white)
        screen.blit(titleTextSurf, (textHorzShift, topCoord))
        if pageNumber == 0:
            One()
        elif pageNumber == 1:
            Two()
        elif pageNumber == 2:
            Three()
        elif pageNumber == 3:
            Four()

        screen.blit(graphicsList[pageNumber], (graphicsHorzShift, graphicsVertShift))

    def Blit(tempText, tempExplanation, tempTop):

        tempSurf = textFont.render(tempText, 1, color.black)
        tempESurf = explanationFont.render(tempExplanation, 1, color.black)

        screen.blit(tempSurf, (textHorzShift, tempTop))
        tempTop += textFontSize + textSpacing
        screen.blit(tempESurf, (textHorzShift, tempTop))
        tempTop += explanationFontSize + textSpacing * 2


        return tempTop

    def Heading(headingText):

        tempTop = textStartVert
        # Heading
        tempSurf = headerFont.render(headingText, 1, color.black)
        screen.blit(tempSurf, (textHorzShift, tempTop))
        tempTop += headerFontSize + headerSpacing

        return tempTop

    def One():

        tempTop = Heading("Experience and Balance (1/4)")

        # Level
        tempText = "Level: " + str(expDict["level"]) + "/" + str(playerInfo.Maximum["level"])
        tempExplanation = "By levelling up using EXP you can earn skill points."
```

```
    tempTop = Blit(tempText, tempExplanation, tempTop)

    # Experience
    tempText = "Experience Points (EXP): " + str(expDict["exp"])
    tempExplanation = "(" + str(expDict["expLevel"]) + " EXP needed for the next level; " ¥
                    + str(expDict["expReq"]) + " left to go)"
    tempExplanation2 = "You can gain EXP from fighting and grinding."

    tempSurf = textFont.render(tempText, 1, color.black)
    tempESurf = explanationFont.render(tempExplanation, 1, color.black)
    tempE2Surf = explanationFont.render(tempExplanation2, 1, color.black)

    screen.blit(tempSurf, (textHorzShift, tempTop))
    tempTop += textFontSize + textSpacing
    screen.blit(tempESurf, (textHorzShift, tempTop))
    tempTop += explanationFontSize + textSpacing
    screen.blit(tempE2Surf, (textHorzShift, tempTop))
    tempTop += explanationFontSize + textSpacing * 2

    # Skill Points
    tempText = "Skill Points: " + str(playerInfo.skillPt)
    tempExplanation = "You can use skill points to level up your skills."

    tempTop = Blit(tempText, tempExplanation, tempTop)

    # Balance
    tempText = "Balance: $" + str(playerInfo.Cash())
    tempExplanation = "You can use cash to buy equipment and potions."

    tempTop = Blit(tempText, tempExplanation, tempTop)

def Two():

    tempTop = Heading("Skills (2/4)")

    explanation = (
        "You will deal a base damage of " + str(playerInfo.fullAP()) + ".",
        "Reduces the damage you take by " + f"{playerInfo.defensePerc() * 100:.3g}" + "%.",
        "Lengthening your answering time by  " + f"{playerInfo.timePerc() * 100:.3g}" + "%.",
        "You have a " + f"{playerInfo.dodgePerc() * 100:.3g}" + "% chance of dodging an attack.",
        "Your maximum health is " + str(playerInfo.fullHP()) + ".",
    )

    for i in range(5):
        skill = playerInfo.SkillIndex[i + 1]
        tempText = playerInfo.SkillName[skill] + ": " ¥
                 + str(playerSkill[skill]) + " / " ¥
                 + str(playerInfo.Maximum["skills"][skill])
        tempExplanation = explanation[i]

        tempTop = Blit(tempText, tempExplanation, tempTop)

def Three():

    tempTop = Heading("Equipment (3/4)")

    explanation = (
        "Reducing the damage you take by " + f"{playerInfo.chestPerc() * 100:.3g}" + "%.",
        "Reducing the damage you take by " + f"{playerInfo.legPerc() * 100:.3g}" + "%.",
        "Strengthing your attacks by " + f"{playerInfo.swordPerc() * 100:.3g}" + "%.",
    )

    for i in range(3):
        equipment = AShop.ShopClass.EquipmentIndex[i + 1]
        tier = equipmentDict[equipment]
        equipementName = AShop.ShopClass.equipmentName(equipment, tier) if tier != 0 else "None"

        tempText = equipment.title() + ": " + equipementName
        tempExplanation = explanation[i]
```

```
            tempTop = Blit(tempText, tempExplanation, tempTop)

    def Four():

        tempTop = Heading("Potions (4/4)")

        explanation = (
            "Regenerates " + f"{HealthPotionPerc * 100:.3g}" + "% of your full health.",
            "Increases your attack power by " + f"{AttackPotionPerc * 100:.3g}" + "% (one turn only).
",
            "Reducing the damage you take by " + f"{DefendPotionPerc * 100:.3g}" + "% (one turn only).
",
        )

        for i in range(3):
            potion = AShop.ShopClass.PotionIndex[i + 1]
            qnty = potionDict[potion]
            potionName = AShop.ShopClass.potionName(potion)

            tempText = potionName + ": " + str(qnty)
            tempExplanation = explanation[i]

            tempTop = Blit(tempText, tempExplanation, tempTop)

    WidthTotal = 29.7
    HeightTotal = 21.0

    # Text Related
    TopCoord = 2.1 / HeightTotal
    TitleFont = 1.4 / HeightTotal
    HeaderFont = 0.9 / HeightTotal
    TextFont = 0.65 / HeightTotal
    ExplanationFont = 0.50 / HeightTotal
    TitleSpacing = 1.6 / HeightTotal
    HeaderSpacing = 0.6 / HeightTotal
    TextSpacing = 0.30 / HeightTotal
    TextHorzShift = 3.2 / WidthTotal

    # Image Related
    GraphicsSize = 9.7 / HeightTotal
    GraphicsVertShift = 5.0 / HeightTotal
    GraphicsHorzShift = 15.4 / WidthTotal
    MenuHorzShift = 2.2 / WidthTotal
    MenuSpacing = 1.1 / WidthTotal
    MenuTextWidth = (5.5 + 1/3) / WidthTotal
    MenuVertShift = 1.9 / HeightTotal
    ScrollerSize = 1.8 / HeightTotal
    MenuTextSize = 0.7 / HeightTotal
    MenuHeight = 1.6 / HeightTotal

    height = screen.get_height()
    width = screen.get_width()

    topCoord = int(TopCoord * height)
    titleFontSize = int(TitleFont * height)
    headerFontSize = int(HeaderFont * height)
    textFontSize = int(TextFont * height)
    explanationFontSize = int(ExplanationFont * height)
    titleSpacing = int(TitleSpacing * height)
    headerSpacing = int(HeaderSpacing * height)
    textSpacing = int(TextSpacing * height)
    textHorzShift = int(TextHorzShift * width)

    graphicsSize = int(GraphicsSize * height)
    graphicsHorzShift = int(GraphicsHorzShift * width)
    graphicsVertShift = int(GraphicsVertShift * height)
    menuHorzShift = int(MenuHorzShift * width)
    menuSpacing = int(MenuSpacing * width)
    menuVertShift = int(MenuVertShift * height)
    scrollerSize = int(ScrollerSize * height)
    menuTextSize = int(MenuTextSize * height)
```

```
    menuHeight = int(MenuHeight * height)
    menuTextWidth = int(MenuTextWidth * width)

    titleFont = pygame.font.Font(DefaultMenuTitleFontString, titleFontSize)
    headerFont = pygame.font.Font(DefaultMenuHeaderFontString, headerFontSize)
    textFont = pygame.font.Font(DefaultMenuTextFontString, textFontSize)
    explanationFont = pygame.font.Font(DefaultMenuExplanationFontString, explanationFontSize)
    menuFont = pygame.font.Font(DefaultMenuMenuFontString, menuTextSize)

    textStartVert = topCoord + titleFontSize + titleSpacing

    pageNumber = 0

    """
    There are 4 pages.
    First page: basic information
    Second page: skills
    Third page: equipment
    Fourth page: potions
    """

    MenuStatOne = pygame.image.load(DefaultMenuStatOne)
    MenuStatTwo = pygame.image.load(DefaultMenuStatTwo)
    MenuStatThree = pygame.image.load(DefaultMenuStatThree)
    MenuStatFour = pygame.image.load(DefaultMenuStatFour)

    graphicsList = [
        MenuStatOne,
        MenuStatTwo,
        MenuStatThree,
        MenuStatFour,
    ]

    for i in range(4):
        graphicsList[i] = pyImage.ScaleGraphics(graphicsList[i], graphicsSize, False, True)

    # Defining the scrolling
    menuRect = [
        (menuHorzShift, height - menuVertShift - menuHeight),
        (menuHorzShift + scrollerSize + menuTextWidth + menuSpacing, height - menuVertShift - menuHe
ight),
        (menuHorzShift + 2 * scrollerSize + 2 * menuTextWidth + menuSpacing * 2, height - menuVertSh
ift - menuHeight),
    ]

    mainMenu = pyMenu.SelectionMenu(DefaultMenuScroller, (menuTextWidth, scrollerSize), alignment=
"center")
    mainMenu.CreateItem("Previous Page", menuFont, menuRect[0])
    mainMenu.CreateItem("Return", menuFont, menuRect[1])
    mainMenu.CreateItem("Next Page", menuFont, menuRect[2])

    titleTextSurf = titleFont.render("Player Statistics", 1, color.black)

    refillRect = (
        0,
        height - menuVertShift - menuHeight,
        width,
        menuVertShift + menuHeight
    )

    Stat(pageNumber)

    # Main Loop:
    exitMenu = False
    while not exitMenu:

        pygame.draw.rect(screen, color.white, refillRect)

        mainMenu.DisplayItem(screen)

        for event in pygame.event.get():
```

```
    Quit(event)

    if event.type == KEYDOWN:
        if event.key == K_RIGHT:
            mainMenu.ScrollItem(1)
        if event.key == K_LEFT:
            mainMenu.ScrollItem(-1)
    counter = mainMenu.Scrolling(event)

    if counter == 2:
        pageNumber = (pageNumber + 1) % 4
        Stat(pageNumber)
    elif counter == 0:
        pageNumber = (pageNumber + 3) % 4
        Stat(pageNumber)
    elif counter == 1:
        exitMenu = True

pygame.display.update()
```

15. APlayer.py

```
"""
APlayer.py
"""

import math
import json

def Initialize():

    return Player({
           "skills" : {
                     "strengthPt" : 0,
                     "defensePt" : 0,
                     "timePt" : 0,
                     "agilityPt" : 0,
                     "healthPt" : 0
                },
           "equipment" : {
                     "chestplate" : 0,
                     "leggings" : 0,
                     "sword" : 0
                },
           "potions" : {
                     "regenPot" : 0,
                     "attackPot" : 0,
                     "defendPot" : 0
                },
           "level" : 1,
           "exp" : 0,
           "skillPt" : 0,
           "bal" : 0,
           "bosslevel" : 1,
           "beat" : False,
       })

def LoadSave(filename):

    try:
        file = open(filename, 'r')
        code = file.read()
        data = json.loads(code)
        return Player(data)
    except:
        print("Save file not found, initializing game with default parameters.")
        return Initialize()

def SaveFile(filename, player):

        file = open(filename, 'w')
        data = player.getDict()
        code = json.dumps(data, indent = 4)
        file.write(code)

class Player:

    SkillIndex = {
        1 : "strengthPt",
        2 : "defensePt",
        3 : "timePt",
        4 : "agilityPt",
        5 : "healthPt"
    }

    SkillName = {
        "strengthPt" : "Strength",
        "defensePt" : "Defense",
        "timePt" : "Time Dilation",
        "agilityPt" : "Agility",
        "healthPt" : "Health"
```

```
}

Maximum = {
    "skills": {
        "strengthPt": 200,
        "defensePt": 200,
        "timePt": 200,
        "agilityPt": 200,
        "healthPt": 200
    },
    "equipment": {
        "chestplate": 10,
        "leggings": 10,
        "sword": 10
    },
    "potions": {
        "regenPot": 0,
        "attackPot": 0,
        "defendPot": 0
    },
    "level": 101,
    "exp": 0,
    "skillPt": 0,
    "bal": 0,
    "bosslevel": 0,
    "beat": False,

}

def __init__(self, data):

    '''
        Reference numbers: (They may or may not be used)

        "skills": {
            "strengthPt": 1,    <- 0
            "defensePt": 1,     <- 1
            "timePt": 1,        <- 2
            "agilityPt": 1,     <- 3
            "healthPt": 1       <- 4
        },
        "equipment": {
            "chestplate": 0,    <- 0
            "leggings": 0,      <- 1
            "sword": 0          <- 2
        },
        "potions": {
            "regenPot": 0,      <- 0
            "attackPot": 0,     <- 1
            "defendPot": 0      <- 2
        },
    '''

    self.skills = data["skills"]
    self.equipment = data["equipment"]
    self.potions = data["potions"]
    self.level = data["level"]
    self.exp = data["exp"]
    self.skillPt = data["skillPt"]
    self.bal = data["bal"]
    self.bosslevel = data["bosslevel"]
    self.beat = data["beat"]

def Skills(self, increments = {}):

    for skillName in increments.keys():
        self.skills[skillName] += increments[skillName]
    return self.skills

def Equipment(self, increments = {}):
```

```
        for equipmentName in increments.keys():
            self.equipment[equipmentName] += increments[equipmentName]
        return self.equipment

    def Potions(self, increments = {}):

        for potionName in increments.keys():
            self.potions[potionName] += increments[potionName]
        return self.potions

    def Experience(self, increment = 0):

        def expLevel(n):
            return math.ceil(100 / (3 ** 0.04 - 1) * (3 ** ((n-1)/25) - 1))

        def expCumulative(n):
            sum = 0
            for i in range(1, n + 1):
                sum += expLevel(i)
            return sum

        def skillPt(n):
            return math.ceil(7.5 * math.e ** (-(n - 1)/150))

        levelUp = False
        self.exp += increment
        while self.exp >= expCumulative(self.level + 1) and self.level + 1 <= self.Maximum["level"]:
            levelUp = True
            self.level += 1
            self.skillPt += skillPt(self.level)

        tempDict = {}
        tempDict["exp"] = self.exp
        tempDict["level"] = self.level
        tempDict["skillPt"] = self.skillPt
        tempDict["expLevel"] = expCumulative(self.level + 1) if self.level + 1 <= self.Maximum["lev
el"] else 0
        tempDict["expReq"] = expCumulative(self.level + 1) - self.exp if self.level + 1 <= self.Maxi
mum["level"] else 0
        tempDict["levelUp"] = levelUp
        return tempDict

    def SkillPt(self, increment = 0):

        self.skillPt += increment
        return self.skillPt

    def getDict(self):
        return {
            "skills" : self.skills,
            "equipment" : self.equipment,
            "potions" : self.potions,
            "level" : self.level,
            "exp" : self.exp,
            "skillPt" : self.skillPt,
            "bal" : self.bal,
            "bosslevel" : self.bosslevel,
            "beat" : self.beat,
        }


    def Beated(self, increment = False):

        self.bosslevel += 1 if increment else 0
        if self.bosslevel >= 3:
            self.bosslevel = 3

        return {
            "bosslevel" : self.bosslevel,
            "beat" : self.beat
        }
```

```
    def Cash(self, increment = 0):
        self.bal += increment
        return self.bal

    def BeatFlag(self):
        self.beat = True

    def fullAP(self):
        return 650 + math.floor(11350 * self.skills["strengthPt"] / 200)
    def fullHP(self):
        return 900 + math.floor(9100 * self.skills["healthPt"] / 200)
    def defensePerc(self):
        return 0.6 * (self.skills["defensePt"] / 200) ** 1.2
    def dodgePerc(self):
        return 0.3 * (self.skills["agilityPt"] / 200)
    def timePerc(self):
        return 2.5 / (1 - math.e ** (-0.76)) * (1 - math.e ** (-0.76 * self.skills["timePt"] / 200))
    def chestPerc(self):
        return 0.3 * self.equipment["chestplate"] / 10
    def legPerc(self):
        return 0.3 * self.equipment["leggings"] / 10
    def swordPerc(self):
        return 0.4 * self.equipment["sword"] / 10


def main():

    myPlayer = LoadSave("TestPlayer.txt")
    myDict = {
           "skills" : {
                     "strengthPt" : myPlayer.skills["strengthPt"],
                     "defensePt" : myPlayer.skills["defensePt"],
                     "timePt" : myPlayer.skills["timePt"],
                     "agilityPt" : myPlayer.skills["agilityPt"],
                     "healthPt" : myPlayer.skills["healthPt"]
                },
           "skillPerc" : {
                     "strengthPt": myPlayer.fullAP(),
                     "defensePt": myPlayer.defensePerc(),
                     "timePt": myPlayer.timePerc(),
                     "agilityPt": myPlayer.dodgePerc(),
                     "healthPt": myPlayer.fullHP()
                },
           "equipment" : {
                     "chestplate" : myPlayer.equipment["chestplate"],
                     "leggings" : myPlayer.equipment["leggings"],
                     "sword" : myPlayer.equipment["sword"]
                },
           "equipmentPerc": {
                     "chestplate": myPlayer.chestPerc(),
                     "leggings": myPlayer.legPerc(),
                     "sword": myPlayer.swordPerc()
                },
           "potions" : {
                     "regenPot" : myPlayer.potions["regenPot"],
                     "attackPot" : myPlayer.potions["attackPot"],
                     "defendPot" : myPlayer.potions["defendPot"]
                },
           "level" : myPlayer.level,
           "exp" : myPlayer.exp,
           "skillPt" : myPlayer.skillPt,
           "bal" : myPlayer.bal,
           "bosslevel" : myPlayer.bosslevel,
           "beat" : myPlayer.beat,
       }
    for i in myDict.keys():
       print(myDict[i])

    SaveFile("TestPlayer.txt", myPlayer)
```

```
if __name__ == "__main__":
    main()
```

16. ASequence.py

```
"""
ASequence.py
"""

from ATextBox import *
import pyAnimation
import GlobalVar

def Epilogue(screen):

    GlobalVar.audio.FeedAudio(GlobalVar.AmbientMusic)

    scene = Scene(screen, nowHeight = 0.25)

    def Zero():
        scene.TextBox.InsertText("~ Epilogue ~", "center", True)

    def One():
        scene.TextBox.InsertText("You have defeated the greatest Mathematicians in all of history.",
 "center")

    def Two():
        scene.TextBox.InsertText("And so, you are recognized as the most influential Mathematician
in the whole world.", "center")

    def Three():
        scene.TextBox.InsertText("You have reached the paramount of human intelligence.", "center")

    def Four():
        scene.TextBox.InsertText("However, as Einstein has said:", "center")
        scene.TextBox.InsertText("Intellectual growth should commence at birth and cease only at de
ath.", "center", True)

    def Five():
        scene.TextBox.InsertText("Do not be satisfied just because you have become the best.", "cent
er")

    def Six():
        scene.TextBox.InsertText("Strive for excellence and perfection, and open new roads that no o
ne have ever found before.", "center")

    def Seven():
        scene.TextBox.InsertText("I believe you can do it.", "center")

    def Eight():
        scene.TextBox.InsertText("Thank you for playing my game, The Becoming of the Mathematician.
", "center")

    def Nine():
        scene.TextBox.InsertText("You may return to the village, save your game, and continue levell
ing yourself until you reach the topmost level. You can also quit the game without saving.")


    FunctionList = [
        Zero,
        One,
        Two,
        Three,
        Four,
        Five,
        Six,
        Seven,
        Eight,
        Nine
    ]

    GraphicsList = [
        "0", "0", "1", "2", "3", "4", "5", "6", "7", "7"
    ]
```

```
scene.ChangeWallpaper("assets/EndingImage/Scene" + "0" + ".jpg", False)
def displayFunction():
    scene.Update()
    scene.Display()

pyAnimation.EnterFade(screen, displayFunction, GlobalVar.CentralClock, GlobalVar.GameFPS)
for num, func in enumerate(FunctionList):

    scene.TextBox.Reset()
    func()
    scene.TextBox.Finalize()

    scene.ChangeWallpaper("assets/EndingImage/Scene" + GraphicsList[num] + ".jpg")

    run = True
    while run:

        for event in pygame.event.get():
            GlobalVar.Quit(event)
            result = scene.TextBox.Event(event)
            if result:
                run = False
                break

        scene.Display()
        LoopBundle()

ret = 0
qui = 1

def Back():
    scene.TextBox.Reset()
    scene.TextBox.InsertMenu("Return to the Village")
    scene.TextBox.InsertMenu("Quit Game")
    scene.TextBox.Finalize()

def Confirm():
    scene.TextBox.Reset()
    scene.TextBox.InsertText("Are you sure?", bold=True)
    nah = scene.TextBox.InsertMenu("No")
    ya = scene.TextBox.InsertMenu("Yes")
    scene.TextBox.Finalize()

    run = True
    while run:

        for event in pygame.event.get():
            GlobalVar.Quit(event)
            result = scene.TextBox.Event(event)
            if result == nah:
                return False
            elif result == ya:
                return True

        scene.Display()
        LoopBundle()

Back()

run = True
while run:

    for event in pygame.event.get():
        GlobalVar.Quit(event)
        result = scene.TextBox.Event(event)
        if result == ret:
            return False
        elif result == qui:
            if Confirm():
                return True
```

```
            else:
                Back()

        scene.Display()
        LoopBundle()


def AdventureModeSequence(screen):

    GlobalVar.audio.FeedAudio(GlobalVar.AmbientMusic)

    scene = Scene(screen, nowHeight = 0.25)

    def Zero():
        scene.TextBox.InsertText("~ The Becoming of a Mathematician ~", "center", True)

    def One():
        scene.TextBox.InsertText("The story takes place in this small, remote village.")
        scene.TextBox.InsertText("Far away from the hustling cities, it was a quiet village.")

    def Two():
        scene.TextBox.InsertText("Then, you were born into this world, just like any other child.")

    def Three():
        scene.TextBox.InsertText("In this remote village, you played around in the fields all day.")
        scene.TextBox.InsertText("Sometimes, you helped your parents with farming.")

    def Four():
        scene.TextBox.InsertText("On one special occasion, you parents brought you out to town.")
        scene.TextBox.InsertText("It was your first time, so you were very excited.")

    def Five():
        scene.TextBox.InsertText("You wandered around this mysterious city, and found a rather mode
rn looking shop.")
        scene.TextBox.InsertText("Out of curiosity, you entered this shop.")

    def Six():
        scene.TextBox.InsertText("You walked around the shop, which was lined up with bookshelves of
 books, and you randomly pulled a book off the shelf.")
        scene.TextBox.InsertText("What you found was a book filled with eccentric and mysterious sym
bols. It was a Math book.")

    def Seven():
        scene.TextBox.InsertText("Your desire to learn resonated, and you asked your parents to buy
you this book.")
        scene.TextBox.InsertText("As you learned and learned, you came across the works of famous Ma
thematicians, and your admiration grew along with the passion to meet them.")

    def Eight():
        scene.TextBox.InsertText("And you finally set your eyes on a final goal: defeat these Mathem
aticians and claim the position of being the best Mathematician.")
        scene.TextBox.InsertText("Just like so, you embarked on a journey to travel around the world
 to defeat these Mathematicians.")

    FunctionList = [
        Zero,
        One,
        Two,
        Three,
        Four,
        Five,
        Six,
        Seven,
        Eight
    ]

    def displayFunction():
        screen.fill((0, 0, 0))
        scene.Display()

    pyAnimation.EnterFade(screen, displayFunction, GlobalVar.CentralClock, GlobalVar.GameFPS)
```

```
    for num, func in enumerate(FunctionList):

        scene.TextBox.Reset()
        func()
        scene.TextBox.Finalize()

        scene.ChangeWallpaper("assets/StoryImage/Scene" + str(num) + ".jpg")

        run = True
        while run:

            for event in pygame.event.get():
                GlobalVar.Quit(event)
                result = scene.TextBox.Event(event)
                if result:
                    run = False
                    break

            scene.Display()
            LoopBundle()


if __name__ == '__main__':
    s = pygame.display.set_mode((1200, 800))
    Epilogue(s)
    AdventureModeSequence(s)
```

17. AShop.py

```
"""
AShop.py
"""


import pyAnimation
import pyImage
import pyMenu
import pygame
from pygame.locals import *
import GlobalVar
import math
import APlayer
import pyFloating
import pyText
import pyInput


class ShopClass:

    EquipmentIndex = {
        1: "chestplate",
        2: "leggings",
        3: "sword"
    }

    PotionIndex = {
        1 : "regenPot",
        2 : "attackPot",
        3 : "defendPot"
    }

    EquipmentPrefix = {
        1 : "Wooden",
        2 : "Carbon",
        3 : "Silver",
        4 : "Iron",
        5 : "Steel",
        6 : "Gold",
        7 : "Diamond",
        8 : "Platinum",
        9 : "Lonsadleite"
    }

    TopEquipmentName = {
        "chestplate" : "Laplace's T-Shirt",
        "leggings" : "Euler's Beach Trunks",
        "sword" : "Newton's Apple Tree Twig'"
    }

    PotionName = {
        "regenPot" : "Potion of Regeneration",
        "attackPot" : "Potion of Braveness",
        "defendPot" : "Potion of Perseverance"
    }

    PotionPrice = {
        "regenPot" : 1000,
        "attackPot" : 4000,
        "defendPot" : 4000
    }

    @staticmethod
    def EquipmentGraphics(equipment, tier):
        if tier < 10:
            if equipment == "chestplate":
                return pygame.image.load(GlobalVar.DefaultChestplate)
            elif equipment == "leggings":
                return pygame.image.load(GlobalVar.DefaultLeggings)
```

```
            elif equipment == "sword":
                return pygame.image.load(GlobalVar.DefaultSword)
            else:
                print("Error occurred in ShopClass.equipmentGraphics.")
        else:
            if equipment == "chestplate":
                return pygame.image.load(GlobalVar.DefaultShirt)
            elif equipment == "leggings":
                return pygame.image.load(GlobalVar.DefaultTrunks)
            elif equipment == "sword":
                return pygame.image.load(GlobalVar.DefaultTwig)
            else:
                print("Error occurred in ShopClass.equipmentGraphics.")

    @staticmethod
    def PotionGraphics(equipment):
        if equipment == "regenPot":
            return pygame.image.load(GlobalVar.DefaultPotion2)
        elif equipment == "attackPot":
            return pygame.image.load(GlobalVar.DefaultPotion1)
        elif equipment == "defendPot":
            return pygame.image.load(GlobalVar.DefaultPotion3)
        else:
            return pygame.image.load(GlobalVar.DefaultPotion4)

    @ staticmethod
    def SkillGraphics():
        return pygame.image.load(GlobalVar.DefaultSkills)

    @staticmethod
    def equipmentPrice(equipment, tier):
        if equipment == "chestplate":
            return math.ceil(8000 * (tier * (1 + tier / 10) ** (tier / 5)))
        elif equipment == "leggings":
            return math.ceil(6000 * (tier * (1 + tier / 10) ** (tier / 5)))
        elif equipment == "sword":
            return math.ceil(10000 * (tier * (1 + tier / 10) ** (tier / 5)))
        else:
            print("Error occurred in ShopClass.equipmentPrice.")

    @staticmethod
    def equipmentName(equipment, tier):
        if tier < 10:
            if equipment in ["chestplate", "leggings", "sword"]:
                return ShopClass.EquipmentPrefix[tier] + " " + equipment.title()
            else:
                print("Error occurred in ShopClass.equipmentName.")
        elif tier == 10:
            return ShopClass.TopEquipmentName[equipment]

    @staticmethod
    def potionName(potion):
        return ShopClass.PotionName[potion]

    @staticmethod
    def potionPrice(potion):
        return ShopClass.PotionPrice[potion]

class ShopMenu:

    GraphicsShift = 0.60 / 6.8
    GraphicsSpacing = 0.60 / 6.8
    ItemMenuShift = 0.2 / 2.1
    GraphicsTextSpacing = 0.2 / 6.8
    ItemMenuTextSpacing = 0.2 / 2.1
    CursorTextRatio = 0.7/(2.1 + 0.7)
    GraphicsTextRatio = 0.7

    def __init__(self, scroller, fontName, dim, exitRect, exitMsg = "Leave shop"):

        self.height = dim[1]
```

```
        self.width = dim[0]
        self.counter = 1

        self.graphicsShift = int(self.GraphicsShift * self.width)
        self.graphicsSpacing = int(self.GraphicsSpacing * self.width)
        self.itemMenuShift = int(self.ItemMenuShift * self.height)
        self.graphicsTextSpacing = int(self.GraphicsTextSpacing * self.height)

        self.graphicsSize = int((self.width - self.graphicsShift*2 - self.graphicsSpacing*2)/3)
        self.graphicsHeight = int(self.height * self.GraphicsTextRatio)
        self.itemMenuHeight = self.height - self.graphicsHeight - self.graphicsTextSpacing
        self.cursorSize = int(self.CursorTextRatio * self.itemMenuHeight)
        self.itemMenuHeight -= int(self.CursorTextRatio * self.itemMenuHeight)
        self.itemMenuWidth = 2 * (self.graphicsShift - self.itemMenuShift) + self.graphicsSize
        self.itemMenuSpacing = int((self.width - 3 * self.itemMenuWidth - 2 * self.itemMenuShift)/
2)
        self.itemMenuTextSpacing = int(self.ItemMenuTextSpacing * self.itemMenuHeight)
        self.itemMenuTextSize = int((self.itemMenuHeight - 2 * self.itemMenuTextSpacing) / 3)

        self.exitRect = exitRect

        self.scroller = scroller
        self.fontName = fontName
        self.itemMenuFont = pygame.font.Font(self.fontName, self.itemMenuTextSize)
        self.graphicsRect = [
            (self.graphicsShift + int(self.graphicsSize/2), int(self.graphicsHeight/2)),
            (self.graphicsShift + self.graphicsSize + self.graphicsSpacing +
             int(self.graphicsSize/2), int(self.graphicsHeight/2)),
            (self.graphicsShift + self.graphicsSize * 2 + self.graphicsSpacing * 2 +
             int(self.graphicsSize/2),  int(self.graphicsHeight/2))
        ]
        self.itemMenuRect = [
            (self.width, self.height),
            (self.itemMenuShift, self.graphicsTextSpacing + self.graphicsHeight),
            (self.itemMenuShift + self.itemMenuWidth + self.itemMenuSpacing,
               self.graphicsTextSpacing + self.graphicsHeight),
            (self.itemMenuShift + self.itemMenuWidth * 2 + self.itemMenuSpacing * 2,
               self.graphicsTextSpacing + self.graphicsHeight)
        ]
        self.exitMenuFont = pygame.font.Font(self.fontName, self.exitRect[3])

        self.itemMenu = pyMenu.SelectionMenu(self.scroller,
                                       (self.itemMenuWidth + 10, self.itemMenuHeight),
                                       alignment = "center", scrollerSize = self.cursorSize)
        for i in range(4):
            self.itemMenu.CreateItem("", self.itemMenuFont, self.itemMenuRect[i],
                                     str(i), multiline=True, scrollerPos="below")
        self.itemMenu.ScrollItem(+1)

        self.exitMenu = pyMenu.SelectionMenu(pygame.transform.rotate(self.scroller, -90),
                                       (self.exitRect[2]+3, self.exitRect[3]),
                                       alignment = "left")
        self.exitMsg = exitMsg
        self.exitMenu.CreateItem(self.exitMsg, self.exitMenuFont, (self.exitRect[:2]), "0")
        self.exitMenu.CreateItem("", self.exitMenuFont, (self.width, self.height), "1")

        self.itemMenu.SetCounter(self.counter)
        self.exitMenu.SetCounter(self.counter)

        self.graphics = [None, None, None]

        self.temp = 1

    def ChangeItem(self, index, text, graphics = None):

        try:
            if graphics:
                self.graphics[index - 1] = pyImage.ScaleGraphics(graphics,  self.graphicsHeight, Fal
se, True)
            else:
                self.graphics[index - 1] = pyImage.ScaleGraphics(pygame.image.load(GlobalVar.Default
```

```
ShopSoldOut),
                                                        self.graphicsHeight, False, True)
            self.itemMenu.ChangeItem(index, text, self.itemMenuFont, self.itemMenuRect[index],
                                     str(index), multiline = True, scrollerPos="below")
        except:
            print("Error occurred in ShopMenu.ChangeItem.")

    def Scroll(self, event, offset):

        if event.type == KEYDOWN:
            if event.key == K_LEFT:
                self.counter += 3
                self.counter %= 4
            elif event.key == K_RIGHT:
                self.counter += 1
                self.counter %= 4
            elif event.key == K_UP:
                if self.counter == 0:
                    self.counter = self.temp
            elif event.key == K_DOWN:
                if self.counter != 0:
                    self.temp = self.counter
                    self.counter = 0
            elif event.key == K_RETURN:
                return self.counter
            self.itemMenu.SetCounter(self.counter)
            self.exitMenu.SetCounter(self.counter)
        else:
            item = self.itemMenu.Clicking(event, offset)
            ex = self.exitMenu.Clicking(event)
            re = False
            if item in range(1, 4):
                re = True
            else:
                if ex == 0:
                    re = True
            self.counter = self.itemMenu.GetSelected() if self.itemMenu.GetSelected() in range(4) el
se self.counter
            self.counter = self.exitMenu.GetSelected() if self.exitMenu.GetSelected() in range(4) el
se self.counter
            self.itemMenu.SetCounter(self.counter)
            self.exitMenu.SetCounter(self.counter)
            if re:
                return self.counter

    def DisplayMenu(self):

        tempSurf = pygame.Surface((self.width, self.height), pygame.SRCALPHA)
        for i in range(3):
            tempRect = self.graphics[i].get_rect()
            tempRect.center = self.graphicsRect[i]
            tempSurf.blit(self.graphics[i], tempRect)
        self.itemMenu.DisplayItem(tempSurf)
        return tempSurf

    def DisplayExit(self, screen):

        self.exitMenu.DisplayItem(screen)

class PaymentBox:

    TextSpacing = 0.2 / 2.1
    HorzShift = 0.2/2.1
    FloaterWidth = 0.6 / 7.1

    def __init__(self, screen, pos, dim, fontName, typingFontName, textBoxWallpaper = None):

        self.height = dim[1]
        self.width = dim[0]
        self.pos = pos
        self.fontName = fontName
```

```
        self.typingFontName = typingFontName
        self.screen = screen

        self.textSpacing = int(self.height * self.TextSpacing)
        self.horzShift = int(self.HorzShift * self.width)
        self.textHeight = int( (self.height - self.textSpacing * 4) / 3 )
        self.textWidth = self.width - 2 * self.horzShift
        self.floaterWidth = int(self.FloaterWidth * self.width)
        self.floaterHeight = self.textHeight + self.textSpacing

        self.totalHeight = self.textHeight * 3 + self.textSpacing * 4

        if textBoxWallpaper:
            self.textBoxWallpaper = pygame.transform.scale(textBoxWallpaper, dim)
        else:
            self.textBoxWallpaper = pygame.transform.scale(GlobalVar.DefaultShopTextboxWallpaper, d
im)

        self.font = pygame.font.Font(self.fontName, min(30, self.textHeight))
        self.typingFont = pygame.font.Font(self.typingFontName, min(30, self.textHeight))

    def PaymentConfirmation(self, confirmationText, textColor = (0, 0, 0), confirmationColor = (0,
0, 0), scroller = None):

        if scroller:
            myScroller = scroller
        else:
            myScroller = GlobalVar.DefaultMenuScroller

        textSurf = self.font.render(confirmationText, 1, textColor)
        blitRect = self.DisplayTextBox(3)

        menu = pyMenu.SelectionMenu(myScroller,
                                    (self.textWidth, self.textHeight),
                                    textShift = self.textSpacing)
        menu.CreateItem("No", self.font, blitRect[1], color = confirmationColor)
        menu.CreateItem("Yes", self.font, blitRect[2], color = confirmationColor)

        # Main loop
        while True:

            for event in pygame.event.get():
                GlobalVar.Quit(event)

                counter = menu.Scrolling(event, self.pos)
                if counter == 0:
                    return False
                elif counter == 1:
                    return True

            tempSurf = self.textBoxWallpaper.copy()
            tempSurf.blit(textSurf, blitRect[0])
            menu.DisplayItem(tempSurf)
            self.screen.blit(tempSurf, self.pos)

            pygame.display.update()
            GlobalVar.CentralClock.tick(30)

    def DisplayTextBox(self, lines):
        if lines == 1:
            return [(self.horzShift, int((self.totalHeight - self.textHeight) / 2))]
        elif lines == 2:
            return [(self.horzShift, int(self.totalHeight/2 - (self.textSpacing / 2) - self.textHeig
ht)),
                    (self.horzShift, int(self.totalHeight/2 + (self.textSpacing / 2)))]
        elif lines == 3:
            return [(self.horzShift, self.textSpacing),
                    (self.horzShift, self.textHeight + self.textSpacing*2),
                    (self.horzShift, self.textHeight*2 + self.textSpacing*3)]
        else:
            print("Error occured in DisplayTextBox.")
```

```
    def PaymentDialogue(self, text, textColor = (0,0,0), floater = None):

        if floater:
            myFloater = pyFloating.Floating(floater,
                                        (self.floaterWidth, self.floaterHeight), GlobalVar.CentralCl
ock)
        else:
            myFloater = pyFloating.Floating(GlobalVar.DefaultShopTextboxFloater,
                                        (self.floaterWidth, self.floaterHeight), GlobalVar.CentralCl
ock)

        lines = pyText.TextWrap(text, self.font, self.textWidth)
        blitRect = self.DisplayTextBox(len(lines))
        linesSurf = pyText.TextRender(lines, self.font, self.textSpacing, color = textColor)

        floaterRect = (self.width - (self.floaterWidth + self.horzShift),
                       self.height - (self.floaterHeight + self.textSpacing))

        # Main loop
        while True:

            for event in pygame.event.get():
                GlobalVar.Quit(event)

                if event.type == KEYDOWN:
                    if event.key == K_RETURN:
                        return

            tempSurf = self.textBoxWallpaper.copy()
            tempSurf.blit(linesSurf, blitRect[0])
            tempSurf.blit(myFloater.Update(), floaterRect)
            self.screen.blit(tempSurf, self.pos)

            pygame.display.update()
            GlobalVar.CentralClock.tick(GlobalVar.GameFPS)


    def Quantity(self, range, msg = "Input quantity:", textColor = (0,0,0), promptColor = (0, 0, 0),
 typeColor = (0, 0, 0)):

        textLines = pyText.TextWrap(msg, self.font, self.textWidth)
        textSurf = pyText.TextRender(textLines, self.font, self.textSpacing, textColor)

        blitRect = self.DisplayTextBox(len(textLines) + 1)
        textPrompt = self.font.render(">>>", 1, promptColor)
        inputBox = pyInput.TextInput(self.typingFont, self.textWidth, GlobalVar.CentralClock, typeC
olor)
        inputBoxRect = (textPrompt.get_width() + self.textSpacing + blitRect[-1][0], blitRect[-1]
[1])

        while True:

            if inputBox.Typing():
                text = inputBox.Text()
                try:
                    if int(text) in range:
                        return int(text)
                except:
                    pass

            tempSurf = self.textBoxWallpaper.copy()
            tempSurf.blit(textSurf, blitRect[0])
            tempSurf.blit(textPrompt, blitRect[-1])
            tempSurf.blit(inputBox.Surface(), inputBoxRect)
            self.screen.blit(tempSurf, self.pos)

            pygame.display.update()
            GlobalVar.CentralClock.tick(30)
```

```
def a_DoShop(screen, playerInfo):

    def GenerateRect(exitMsg):

        exitRectTextWidth = pygame.font.Font(GlobalVar.BattleFontString, exitRectHeight).size(exit
Msg)[0]
        return (int(screenWidth / 2 - exitRectHeight - exitRectTextWidth / 2),
                  topShift + menuHeight + int((bottomShift - exitRectHeight) / 2),
                  exitRectTextWidth, exitRectHeight)

    def Title(title):

        balance = int(playerInfo.Cash())
        balanceText = "Balance: " + str(balance)

        titleSurf = titleFont.render(title, 1, GlobalVar.color.black)
        balanceSurf = balanceFont.render(balanceText, 1, GlobalVar.color.black)

        screen.blit(titleSurf, titleRect)
        screen.blit(balanceSurf, balanceRect)

    def ArmourShop(screen, playerInfo):

        def Update(index):
            equipment = ShopClass.EquipmentIndex[index]
            nowtier = playerInfo.Equipment()[equipment]
            tier = nowtier + 1

            if tier <= 10:
                armourMenu.ChangeItem(index,
                                      ShopClass.equipmentName(equipment, tier) + "¥n" # text
                                      + '$' + str(ShopClass.equipmentPrice(equipment, tier)),
                                      ShopClass.EquipmentGraphics(equipment, tier)) # graphics
            else:
                armourMenu.ChangeItem(index, "Sold Out!")

        aExitMsg = "Leave Shop"
        armourMenu = ShopMenu(GlobalVar.DefaultShopScroller, GlobalVar.BattleFontString,
                              (screenWidth, menuHeight), GenerateRect(aExitMsg),
                              aExitMsg)

        for index in range(1, 4):
            Update(index)

        while True:

            screen.fill(GlobalVar.color.white)
            Title("Armour and Weapons Shop")
            screen.blit(armourMenu.DisplayMenu(), (0, topShift))
            armourMenu.DisplayExit(screen)

            for event in pygame.event.get():
                GlobalVar.Quit(event)
                counter = armourMenu.Scroll(event, (0, topShift))

                if counter == 0:
                    # Leave Shop
                    return
                elif counter in range(1, 4):
                    equipment = ShopClass.EquipmentIndex[counter]
                    tier = playerInfo.Equipment()[equipment]
                    # Can the item be purchased?
                    if tier + 1 <= 10:
                        # Payment Process begins
                        textBox = PaymentBox(screen, (textBoxHorzShift, topShift + menuHeight + textBo
xVertShift),
                                             (textBoxWidth, textBoxHeight),
                                             GlobalVar.DefaultShopTextboxTextFontString,
                                             GlobalVar.DefaultShopTextboxInputFontString)

                        nowtier = playerInfo.Equipment()[equipment]
```

```
                    tier = nowtier + 1
                    name = ShopClass.equipmentName(equipment, tier)
                    price = ShopClass.equipmentPrice(equipment, tier)

                    # Does the player have enough money?
                    if playerInfo.Cash() >= price:
                        confirmation = textBox.PaymentConfirmation("Confirm Purchase: " + name +
"?",
                                                                   GlobalVar.color.blue)
                        if confirmation:
                            # The player buys it
                            playerInfo.Cash(-1 * price)
                            playerInfo.Equipment({equipment : 1})
                            textBox.PaymentDialogue("You have successfully purchased a " + name + ".
")
                            Update(counter)
                        else:
                            # The player does not buy it
                            textBox.PaymentDialogue("Payment cancelled.", GlobalVar.color.blue)

                    # The player does not have enough money
                    else:
                        textBox.PaymentDialogue("You do not have enough cash.")
                else:
                    pass

            GlobalVar.CentralClock.tick(30)
            pygame.display.update()

    def PotionShop(screen, playerInfo):

        def Update(index):
            potion = ShopClass.PotionIndex[index]
            potionMenu.ChangeItem(index,
                                  ShopClass.potionName(potion) + "¥n"  # text
                                  + '$' + str(ShopClass.potionPrice(potion)),
                                  ShopClass.PotionGraphics(potion))  # graphics

        aExitMsg = "Leave Shop"
        potionMenu = ShopMenu(GlobalVar.DefaultShopScroller, GlobalVar.BattleFontString,
                              (screenWidth, menuHeight), GenerateRect(aExitMsg),
                              aExitMsg)

        for index in range(1, 4):
            Update(index)

        while True:

            screen.fill(GlobalVar.color.white)
            Title("Potion Shop")
            screen.blit(potionMenu.DisplayMenu(), (0, topShift))
            potionMenu.DisplayExit(screen)

            for event in pygame.event.get():
                GlobalVar.Quit(event)

                counter = potionMenu.Scroll(event, (0, topShift))

                if counter == 0:
                    # Leave Shop
                    return
                elif counter in range(1, 4):
                    # Payment Process begins
                    textBox = PaymentBox(screen, (textBoxHorzShift, topShift + menuHeight + textBoxVe
rtShift),
                                         (textBoxWidth, textBoxHeight),
                                         GlobalVar.DefaultShopTextboxTextFontString,
                                         GlobalVar.DefaultShopTextboxInputFontString)

                    # Ask the player for quantity
                    potion = ShopClass.PotionIndex[counter]
```

```
                name = ShopClass.potionName(potion)
                price = ShopClass.potionPrice(potion)

                qnty = textBox.Quantity(range(0, 100),
                                        "How many of the " + name + " would you like to buy? (0 to 99)
")

                # Does the player want to actually buy it?
                if qnty != 0:

                    totalPrice = qnty * price

                    # Does the player have enough money?
                    if playerInfo.Cash() >= totalPrice:
                        confirmation = textBox.PaymentConfirmation("Confirm Purchase: " + name
                                                                   + ", Quantity: " + str(qnty) + "?",
                                                                   GlobalVar.color.blue)

                        if confirmation:
                            playerInfo.Cash(-1 * totalPrice)
                            playerInfo.Potions({potion : qnty})
                            textBox.PaymentDialogue("Payment successful.", GlobalVar.color.blue)
                        else:
                            textBox.PaymentDialogue("Payment cancelled.", GlobalVar.color.blue)
                    else:
                        textBox.PaymentDialogue("You do not have enough cash.", GlobalVar.color.bl
ue)
                else:
                    textBox.PaymentDialogue("Payment cancelled.", GlobalVar.color.blue)


            pygame.display.update()
            GlobalVar.CentralClock.tick(30)

    def SkillShop(screen, playerInfo):

        def Title(title):

            balanceText = "Skill Points: " + str(playerInfo.SkillPt())

            titleSurf = titleFont.render(title, 1, GlobalVar.color.black)
            balanceSurf = balanceFont.render(balanceText, 1, GlobalVar.color.black)

            screen.blit(titleSurf, titleRect)
            screen.blit(balanceSurf, balanceRect)

        STextHeight = 0.4
        SExplanationHeight = 0.15
        SExplanationSpacing = 0.03
        STextSpacing = 0.5
        SHeightTotal = STextHeight * 3 + STextSpacing * 4
        STextShift = 0.3
        SWidthTotal = 6.3

        sTextHeight = int(STextHeight / SHeightTotal * menuHeight)
        sExplanationHeight = int(SExplanationHeight / SHeightTotal * menuHeight)
        sExplanationSpacing = int(SExplanationSpacing / SHeightTotal * menuHeight)
        sTextSpacing = int(STextSpacing / SHeightTotal * menuHeight)
        sTextShift = int(STextShift / SWidthTotal * screenWidth)
        sHalfWidth = int(screenWidth / 2)
        sTextWidth = sHalfWidth - 2 * sTextShift
        sFont = pygame.font.Font(GlobalVar.DefaultShopFontString, exitSize)
        sEFont = pygame.font.Font(GlobalVar.DefaultMenuExplanationFontString, sExplanationHeight)

        exitMsg = "Leave Shop"
        exitRect = GenerateRect(exitMsg)

        textRect = [
            exitRect,
            (sTextShift, topShift + sTextSpacing),
            (sTextShift, topShift + sTextSpacing * 2 + sTextHeight),
```

```
            (sTextShift, topShift + sTextSpacing * 3 + sTextHeight * 2),
            (sTextShift + sHalfWidth, topShift + sTextSpacing),
            (sTextShift + sHalfWidth, topShift + sTextSpacing * 2 + sTextHeight)
        ]

        explanationRect = [
            (sTextShift + sTextHeight, topShift + sTextSpacing + sExplanationSpacing + sTextHeight),
            (sTextShift + sTextHeight, topShift + sTextSpacing * 2 + sTextHeight + sExplanationSpaci
ng + sTextHeight),
            (sTextShift + sTextHeight, topShift + sTextSpacing * 3 + sTextHeight * 2 + sExplanationSp
acing + sTextHeight),
            (sTextShift + sHalfWidth + sTextHeight, topShift + sTextSpacing + sExplanationSpacing +
sTextHeight),
            (sTextShift + sHalfWidth + sTextHeight, topShift + sTextSpacing * 2 + sTextHeight + sExpl
anationSpacing + sTextHeight)
        ]

        skillDict = playerInfo.Skills()
        menu = pyMenu.SelectionMenu(GlobalVar.MenuArrow,
                                    (sTextWidth, sTextHeight),
                                    textShift = sTextSpacing, scrollerSize=exitSize)


        menu.CreateItem(exitMsg, sFont, textRect[0])
        for index in range(1, 6):
            skill = APlayer.Player.SkillIndex[index]
            menu.CreateItem("[" + str(skillDict[skill]) + "/200] " + APlayer.Player.SkillName[skil
l],
                            sFont, textRect[index])


        explanationSurf = [0 for i in range(5)]

        def UpdateStatement():

            explanation = (
                "You will deal a base damage of " + str(playerInfo.fullAP()) + ".",
                "Reduces the damage you take by " + f"{playerInfo.defensePerc() * 100:.3g}" + "%.",
                "Lengthening your answering time by  " + f"{playerInfo.timePerc() * 100:.3g}" + "%.",
                "You have a " + f"{playerInfo.dodgePerc() * 100:.3g}" + "% chance of dodging an attac
k.",
                "Your maximum health is " + str(playerInfo.fullHP()) + ".",
            )

            for i in range(5):
                lines = pyText.TextWrap(explanation[i], sEFont, int(sHalfWidth - 2 * sTextShift - sTe
xtHeight))
                explanationSurf[i] = pyText.TextRender(lines, sEFont, SExplanationSpacing)

        UpdateStatement()

        while True:

            screen.fill(GlobalVar.color.white)
            Title("Skill Shop")
            menu.DisplayItem(screen)
            for i in range(5):
                screen.blit(explanationSurf[i], explanationRect[i])
            pygame.display.update()

            for event in pygame.event.get():
                GlobalVar.Quit(event)

                counter = menu.Scrolling(event)

                if counter == 0:
                    # Leave Shop
                    return
                elif counter in range(1, 6):
                    # Payment Process begins
                    textBox = PaymentBox(screen, (textBoxHorzShift, topShift + menuHeight + textBoxVe
```

```
rtShift),
                                    (textBoxWidth, textBoxHeight),
                                    GlobalVar.DefaultShopTextboxTextFontString,
                                    GlobalVar.DefaultShopTextboxInputFontString)

                    # Ask the player for quantity
                    skill = APlayer.Player.SkillIndex[counter]
                    name = APlayer.Player.SkillName[skill]
                    skillPt = playerInfo.SkillPt()
                    skillLvl = playerInfo.Skills()[skill]

                    # Does the player have any points?
                    if skillPt > 0:
                        # Can the player put points?
                        if skillLvl < 200:
                            maxPut = min(skillPt, 200 - skillLvl)
                            qnty = textBox.Quantity(range(0, maxPut + 1),
                                                    "How many points would you like to put into "
                                                    + APlayer.Player.SkillName[skill] + "? (0 - " + str(ma
xPut) + ")")

                            # Confirmation
                            if qnty != 0:
                                confirmation = textBox.PaymentConfirmation("Confirm: dedicate " + str(q
nty)
                                                                           + " " + ("points" if qnty > 1 else "
point") +
                                                                           " into " + name + "?", GlobalVar.co
lor.blue)
                                if confirmation:
                                    playerInfo.SkillPt(-1 * qnty)
                                    newSkill = playerInfo.Skills({skill : qnty})
                                    textBox.PaymentDialogue("Payment successful! Your " + name + " skill
 is now "
                                                            + "level " + str(newSkill[skill]) + "!")

                                    # Redefine Menu
                                    skill = APlayer.Player.SkillIndex[counter]
                                    menu.ChangeItem(counter, "[" + str(skillDict[skill]) + "/200] " +
                                                    APlayer.Player.SkillName[skill],
                                                    sFont, textRect[counter])
                                    UpdateStatement()

                                else:
                                    textBox.PaymentDialogue("Payment cancelled.")
                            else:
                                textBox.PaymentDialogue("Payment cancelled.")
                        else:
                            textBox.PaymentDialogue("You have already maxed out " + name + "!")
                    else:
                        textBox.PaymentDialogue("You don't have any skill points!" +
                                                "¥nEarn some from grinding or fighting Mathematicians.")

            GlobalVar.CentralClock.tick(30)

    # ******************End of Module Definition*****************************

    TopShift = 0.8
    MenuHeight = 2.1
    BottomShift = 1.4
    HeightTotal = TopShift + MenuHeight + BottomShift
    TitleSize = 0.30
    BalanceSize = 0.2
    TextBoxHeight = 1
    TextBoxWidth = 5.4
    WidthTotal = 6.8
    HorzShift = 0.4
    BalanceShift = WidthTotal - 2.0

    screenHeight = screen.get_height()
    screenWidth = screen.get_width()
```

```
    topShift = int(TopShift / HeightTotal * screenHeight)
    menuHeight = int(MenuHeight / HeightTotal * screenHeight)
    bottomShift = int(BottomShift / HeightTotal * screenHeight)
    textBoxHeight = int(TextBoxHeight / HeightTotal * screenHeight)
    textBoxWidth = int(TextBoxWidth / WidthTotal * screenWidth)
    textBoxVertShift = int((bottomShift - textBoxHeight) / 2)
    textBoxHorzShift = int((screenWidth - textBoxWidth) / 2)
    titleSize = int(TitleSize / HeightTotal * screenHeight)
    titleShift = int( (topShift - titleSize) / 2 )
    balanceSize = int(BalanceSize / HeightTotal * screenHeight)
    balanceHorzShift = int(BalanceShift / WidthTotal * screenWidth)
    balanceVertShift = int( (topShift - balanceSize) / 2 )
    horzShift = int(HorzShift / WidthTotal * screenWidth)

    titleFont = pygame.font.Font(GlobalVar.DefaultShopTitleFontString, titleSize)
    balanceFont = pygame.font.Font(GlobalVar.DefaultShopBalanceFontString, balanceSize)
    titleRect = (horzShift, titleShift)
    balanceRect = (balanceHorzShift, balanceVertShift)

    exitMsg = "Leave the Shops"
    exitRectHeight = int(0.0388 * screenHeight)
    exitRect = GenerateRect(exitMsg)
    exitSize = exitRect[3]

    mainMenu = ShopMenu(GlobalVar.DefaultShopScroller, GlobalVar.BattleFontString, (screenWidth, m
enuHeight), exitRect, exitMsg)

    mainMenu.ChangeItem(1, "Armour and Weapon Shop", ShopClass.EquipmentGraphics("chestplate", 1))
    mainMenu.ChangeItem(2, "Potion Shop", ShopClass.PotionGraphics("else"))
    mainMenu.ChangeItem(3, "Skill Shop", ShopClass.SkillGraphics())

    tempSurf = mainMenu.DisplayMenu()
    def displayFunction():
        screen.fill(GlobalVar.color.white)
        Title("The Shops")
        screen.blit(tempSurf, (0, topShift))
        mainMenu.DisplayExit(screen)

    GlobalVar.audio.FeedAudio(GlobalVar.ShopMusic)

    pyAnimation.EnterFade(screen, displayFunction, GlobalVar.CentralClock, GlobalVar.GameFPS)

    while True:

        tempSurf = mainMenu.DisplayMenu()
        displayFunction()

        for event in pygame.event.get():
            GlobalVar.Quit(event)
            counter = mainMenu.Scroll(event, (0, topShift))
            if counter == 0:
                return
            elif counter == 1:
                # Armour and Weapon Shop
                ArmourShop(screen, playerInfo)
            elif counter == 2:
                # Potion Shop
                PotionShop(screen, playerInfo)
            elif counter == 3:
                # Skill Shop
                SkillShop(screen, playerInfo)


        GlobalVar.LoopBundle()


def main():

    screen = pygame.display.set_mode((1280, 800))
    menu = ShopMenu(GlobalVar.DefaultShopScroller, GlobalVar.BattleFontString, (1280, 500), (30, 6
```

```
00, 900, 60))
    # Accepted range: 1280 to 400 ~ 500
    menu.ChangeItem(1, "Hikari¥n$40000", pygame.image.load("assets/Hikari.png"))
    menu.ChangeItem(2, "Tairitsu¥n$60000", pygame.image.load("assets/Tairitsu.png"))
    menu.ChangeItem(3, "Magnificent Tairitsu¥n$400000", pygame.image.load("assets/Tairitsu3.jpg"))

    while True:
        screen.fill(GlobalVar.color.white)
        for event in pygame.event.get():
            GlobalVar.Quit(event)
            k = menu.Scroll(event)

        screen.blit(menu.DisplayMenu(), (0, 0))
        menu.DisplayExit(screen)
        pygame.display.update()

if __name__ == "__main__":
    main()
```

18. ATextBox.py

```
"""
ATextBox.py
"""


import pygame
import pyText
import pyMenu
import pyImage
import pyFloating
from pygame.locals import *
from GlobalVar import *


class TextBox:
    # Constants
    TotalHeight = 0.4
    BoxShift = 0.05 / TotalHeight  # Box Text shift is 0.05 height
    BoxSpacing = 0.02 / TotalHeight  # Box Text spacing is 0.02 height
    TextShift = 0.03  # Distance between special objects and text
    TextSize = 0.03 / TotalHeight
    FloatWidth = 0.07
    FloatHeight = 0.06 / TotalHeight

    SelectionWidth = 1 - 2 * BoxShift

    def __init__(self, size, textBoxFrame=None, menuAlignment="left", nowHeight = 0.4):

        self.height = int(size[1])
        self.width = int(size[0])
        self.menuAlignment = menuAlignment

        # Display Constants
        self.boxShift = int(self.BoxShift * self.height)
        self.horzBoxShift = int(self.boxShift / nowHeight * self.TotalHeight)
        self.boxSpacing = int(self.BoxSpacing * self.height)
        self.textShift = int(self.TextShift * self.width)
        self.textSize = int(self.TextSize * self.height * self.TotalHeight / nowHeight)

        # Floater (The arrow that prompts the player to press Enter)
        self.floatWidth = int(self.FloatWidth * self.width)
        self.floatHeight = int(self.FloatHeight * self.height * self.TotalHeight / nowHeight)
        self.floaterRect = (self.width - self.floatWidth - self.horzBoxShift * 2,
                            self.height - self.floatHeight - self.boxShift)
        self.floater = pyFloating.Floating(FloaterArrow, (self.floatWidth, self.floatHeight), Centr
alClock)

        # Font
        self.GameFont = pygame.font.Font(GameFont, self.textSize)
        self.GameBoldFont = pygame.font.Font(GameFont, self.textSize)
        self.GameBoldFont.set_bold(True)

        # Selection Menus
        self.selectionHeight = self.textSize
        self.selectionWidth = int(self.SelectionWidth * self.width)
        self.selectionSurface = (self.selectionWidth, self.selectionHeight)

        self.BlitList = []
        self.MenuList = []
        self.TextList = []
        self.counter = 0
        self.menuCounter = 0
        self.Menu = None

        if textBoxFrame == None:
            self.textBoxFrame = pygame.transform.scale(DefaultTextBox, (self.width, self.height))
        else:
            self.textBoxFrame = pygame.transform.scale(textBoxFrame, (self.width, self.height))
```

```
    def InsertText(self, text, alignment="left", bold=False, color=(0, 0, 0)):

        lines = pyText.TextWrap(text, self.GameBoldFont if bold else self.GameFont, self.width - se
lf.horzBoxShift * 2)
        for i in lines:
            self.TextList.append([self.counter, i, alignment, bold, color])
            self.counter += 1

    def InsertMenu(self, text, color=(0, 0, 0)):

        self.MenuList.append([self.counter, text, color])
        self.counter += 1
        self.menuCounter += 1
        return self.menuCounter - 1

    def Finalize(self):

        if self.MenuList:
            self.Menu = pyMenu.SelectionMenu(DefaultMenuScroller, self.selectionSurface)

        topCoord = int((self.height - (self.boxSpacing * (self.counter - 1) + self.textSize * self.c
ounter)) / 2)

        for i in self.TextList:
            height = int((self.boxSpacing + self.textSize) * (i[0]))
            if i[3]:
                surf = self.GameBoldFont.render(i[1], 1, i[4])
            else:
                surf = self.GameFont.render(i[1], 1, i[4])

            if i[2] == "left":
                rect = (self.horzBoxShift, topCoord + height)
            elif i[2] == "right":
                rect = (self.width - self.horzBoxShift - surf.get_width(), topCoord + height)
            else:
                rect = (int(self.width - surf.get_width()) / 2,
                        topCoord + height)
            self.BlitList.append((surf, rect))

        if self.MenuList:

            for i in self.MenuList:
                height = int((self.boxSpacing + self.textSize) * (i[0]))
                if self.menuAlignment == "left":
                    rect = (self.horzBoxShift, topCoord + height)
                elif self.menuAlignment == "right":
                    rect = (self.width - self.horzBoxShift - self.selectionWidth - self.selectionHeig
ht,
                            topCoord + height)
                else:
                    rect = (int(self.width - self.selectionWidth - self.selectionHeight) / 2,
                            topCoord + height)
                self.Menu.CreateItem(i[1], self.GameFont, rect)

    def Blit(self):

        tempSurf = self.textBoxFrame.copy()
        for i in self.BlitList:
            tempSurf.blit(i[0], i[1])

        if self.Menu:
            self.Menu.DisplayItem(tempSurf)
        else:
            tempSurf.blit(self.floater.Update(), self.floaterRect)

        return tempSurf

    def Event(self, eventHa, offset = (0, 0)):

        if self.Menu:
            return self.Menu.Scrolling(eventHa, offset)
```

```
        else:
            if eventHa.type == KEYDOWN:
                if eventHa.key == K_RETURN:
                    return True
                else:
                    return False

    def Reset(self):

        self.BlitList = []
        self.MenuList = []
        self.TextList = []
        self.counter = 0
        self.Menu = None
        self.menuCounter = 0


class Scene:
    # Constants
    DisplayHeight = 0.6  # Graphics take 40% of the screen height
    BoxHeight = 1 - DisplayHeight

    # Initiator Function
    def __init__(self, screen, textBoxFrame=None, nowHeight = 0.4):

        # Copying Arguments
        self.screen = screen
        self.height = screen.get_height()
        self.width = screen.get_width()

        # Screen planning
        self.displayHeight = int((1-nowHeight) * self.height)
        self.boxHeight = self.height - self.displayHeight

        self.TextBox = TextBox((self.width, self.boxHeight), textBoxFrame, nowHeight = nowHeight)
        self.displayWallpaper = None

    def Display(self):
        self.screen.blit(self.TextBox.Blit(), (0, self.displayHeight))

    def ChangeWallpaper(self, Wallpaper, update = True):

        self.displayWallpaper = pyImage.Wallpaper(pygame.image.load(Wallpaper), self.screen.get_si
ze())
        if update:
            self.Update()

    def Update(self):
        self.screen.blit(self.displayWallpaper, (0, 0))


class PopUp:

    BoxHeight = 0.4
    BoxWidth = 0.5

    # Initiator Function
    def __init__(self, screen, textBoxFrame=None):
        # Copying Arguments
        self.screen = screen
        self.height = screen.get_height()
        self.width = screen.get_width()

        self.boxHeight = int(self.BoxHeight * self.height)
        self.boxWidth = int(self.BoxWidth * self.width)
        self.BoxRect = (int((self.width - self.boxWidth) / 2), int((self.height - self.boxHeight))
/ 2)
        self.TextBox = TextBox((self.boxWidth, self.boxHeight), textBoxFrame)

    def Offset(self):
        return self.BoxRect
```

```
    def Start(self):
        surf = pygame.Surface((self.screen.get_size()))
        surf.set_alpha(150)
        self.screen.blit(surf, (0, 0))

    def Blit(self):
        self.screen.blit(self.TextBox.Blit(), self.BoxRect)


if __name__ == '__main__':
    s = pygame.display.set_mode((1200 // 2, 800 // 2))
    k = TextBox((1200 // 2, 800 // 2 * 0.4))
    k.InsertText("Fkfdsf")
    k.Finalize()
    while True:

        for event in pygame.event.get():
            Quit(event)

            result = k.Event(event)
            if result:
                print("HAhsdklfhskf")

        s.blit(k.Blit(), (0, 0))
        LoopBundle()
```

19. ATutor.py

```
"""
ATutor.py
"""


import os
import GlobalVar

def a_tutor(playerInfo):
    diff = playerInfo.Beated(False)["bosslevel"] - 1
    beat = playerInfo.Beated(False)["beat"]

    if beat:
        os.startfile(os.path.abspath(GlobalVar.GuidePath))
    else:
        os.startfile(os.path.abspath(GlobalVar.GuideFilename[diff]))
```

20. AVillage.py

```
"""
AVillage.py
"""


import ATutor
import pyMenu
import pygame
import GlobalVar
import AGrinding
import ABossBattle
import AShop
import AMenu
import pyAnimation
import ATextBox
import pyImage
import mDebug


def a_Village(screen, playerInfo, sequence = True):

    def displayFunction():
        screen.fill((255, 255, 255))
        menu.Blit()

    menu = pyMenu.FrameMenu(screen)

    Wallpaper = pygame.image.load(GlobalVar.DefaultVillageWallpaper)
    menu.AddGraphics(Wallpaper)

    menu.InsertText("Village", GlobalVar.color.black, Title = True)

    menu.InitiateMenu()

    shops = menu.InsertMenu("The Shops")
    grind = menu.InsertMenu("Grinding")
    fight = menu.InsertMenu("Journey On...")

    menu.InsertText("", Enter = True)
    guide = menu.InsertMenu("Math Guide")
    gmenu = menu.InsertMenu("Game Menu")


    displayFunctionSurf = menu.DisplayFunctionSurf()
    def display2Function():
        screen.fill((255, 255, 255))
        screen.blit(displayFunctionSurf, (0, 0))

    pyAnimation.EnterFade(screen, display2Function, GlobalVar.CentralClock, GlobalVar.GameFPS)

    popUp = ATextBox.PopUp(screen)

    functionList = [
        lambda : popUp.TextBox.InsertText("Welcome to the Village!", "center", bold = True),
        lambda : popUp.TextBox.InsertText("Your goal is to defeat Mathematicians by intellectual ba
ttles: "
                                          "answering the math questions that they ask you. You can fight
"
                                          "Mathematicians by selecting “Journey On...”.", "center"),
        lambda : popUp.TextBox.InsertText("You should practice and become stronger before fighting
them. "
                                          "To do that, select “Grinding”. This mode allows you to battle "
                                          "people.", "center"),
        lambda : popUp.TextBox.InsertText("You can gain skill points and money by fighting Mathemati
cians "
                                          "and by Grinding. Spend them at “The Shops” to strength your att
ack"
                                          " or defense skills.", "center"),
        lambda : popUp.TextBox.InsertText("For other settings, please access “Game Menu”. You can ch
```

```
eck your "
                                    "game statistics, save your game, or quit the game through ther
e.",
                                    "center"),
        lambda: popUp.TextBox.InsertText("Game Statistics in the ¥"Game Menu¥" has detailed explana
tion about what "
                                    "equipment and potions do. If you are unsure about them, "
                                    "be sure to visit it.", "center"),
        lambda : popUp.TextBox.InsertText("Have fun!", "center", bold = True)
    ] if sequence else [
        lambda: popUp.TextBox.InsertText("Welcome back!", "center", bold=True)
    ]

    for i in functionList:
        popUp.TextBox.Reset()
        i()
        popUp.TextBox.Finalize()
        displayFunction()
        popUp.Start()

        run = True
        while run:

            for event in pygame.event.get():
                GlobalVar.Quit(event)
                result = popUp.TextBox.Event(event, popUp.Offset())
                if result:
                    run = False
            popUp.Blit()
            GlobalVar.LoopBundle()

    GlobalVar.audio.FeedAudio(GlobalVar.VillageMusic)

    # menu.menu.SetCounter(1)

    exitFlag = False
    while not exitFlag:

        displayFunction()

        for event in pygame.event.get():

            temp = mDebug.ReloadPlayer(event)
            if temp:
                playerInfo = temp
                temp = None

            GlobalVar.Quit(event)

            choice = menu.Scrolling(event)
            if choice == shops:
                # Shops
                AShop.a_DoShop(screen, playerInfo)
            elif choice == grind:
                # Grinding
                AGrinding.a_DoGrinding(screen, playerInfo)
            elif choice == fight:
                # Fight Boss
                exitFlag = ABossBattle.a_DoBoss(screen, playerInfo)
            elif choice == gmenu:
                # Menu
                exitFlag = AMenu.a_Menu(screen, playerInfo)
            elif choice == guide:
                # Guide
                ATutor.a_tutor(playerInfo)

            if choice != None and choice != 3 and not exitFlag:
                GlobalVar.audio.FeedAudio(GlobalVar.VillageMusic)
                displayFunctionSurf = menu.DisplayFunctionSurf()
                pyAnimation.EnterFade(screen, displayFunction, GlobalVar.CentralClock, GlobalVar.Gam
eFPS)
```

```
        GlobalVar.LoopBundle()

    pyAnimation.FadeOut(screen, GlobalVar.CentralClock, GlobalVar.GameFPS)

def main():
    import pygame
    import APlayer
    screen = pygame.display.set_mode((1280, 800))
    playerInfo = APlayer.LoadSave("TestPlayer.txt")
    a_Village(screen, playerInfo, False)

if __name__ == "__main__":
    main()
```

21. mMultiplayerBattle.py

```
"""
mMultiplayerBattle.py
"""

import pygame

import GlobalVar
import Main_Battle as Battle
import mDebug
import mMultiplayerClient
import mQuestions
import pyImage
import pyInput
import pyMenu
import pyText
import pyTimer


class OnlineBattle(Battle.MultiplayerBattle):

    # Display Constants
    DefaultLogFontColor = pygame.Color("darkred")
    GameFPS = 60
    DefaultWaitTime = 3 # seconds
    DefaultPenaltyTime = 10 # seconds
    DefaultPastTime = 3 # seconds
    DefaultStartTime = 5 # seconds

    def __init__(self, screen, IP, port, name):

        self.DefaultPlayer1 = pygame.image.load(GlobalVar.ProfileImage)
        self.DefaultPlayer2 = pygame.image.load(GlobalVar.Player2Image)

        super().__init__(screen, BattleGraphics=GlobalVar.DefaultMultiOnline, LightName = True)

        self.logTextSize = int(self.textSize)
        self.logHeight = self.boxSpacing * 2 + self.logTextSize
        self.logWidth = int(self.width * 5 / 7)
        self.logSpacing = int(self.boxSpacing // 2)
        self.logPos = (int((self.width - self.logWidth) // 2), self.displayHeight - self.logHeight)
        self.logBox = pygame.transform.scale(GlobalVar.DefaultLogBox, (self.logWidth, self.logHeig
ht))
        self.logFont = pygame.font.Font(GlobalVar.BlockyFont, self.logTextSize)
        self.logCenter = (int(self.width // 2), self.displayHeight - self.logSpacing - int(self.log
TextSize // 2))

        self.AttackStm = ""

        self.threadGo = True
        self.displayUpdate = True
        self.state = "Game"

        self.run = True

        self.client = mMultiplayerClient.Client(IP, port, name)

        self.p = self.client.playerNo
        self.clock = pygame.time.Clock()

        self.timeOutNo = 0
        self.timeOutMax = 20

    """from memory_profiler import profile
    @profile"""
    def Main(self):

        # filename="logging" + str(self.p), , filemode="w"

        # Opening new client
```

```
        client = self.client
        p = self.p

        menu = pyMenu.FrameMenu(self.screen)
        menu.InsertText("Online Multiplayer Battle", Title=True)
        menu.InsertText("Waiting for the game to start...", Header=True)

        menu.InitiateMenu()
        menu.InsertMenu("Return")

        # Waiting for the game to start
        run = True
        while run:
            self.screen.fill(pygame.Color("white"))
            game = client.getStatus()

            if self.PendingStart(menu):
                # The player would like to quit the game.
                return

            # Testing whether the game has started
            if game.GameState() == "GAME":
                run = False
                break

        # The game has started
        # Main: retrieving the game status, terminating whenever necessary
        # Thread: displaying the game
        #   - The battle
        #   - Text box
        #   - Status bar
        # The thread is a daemon thread which will be terminated
        # when run is false. An exception can also be raised.
        # Game has started, reuse previous game
        self.playerSprite = game.p1 if p == "1" else game.p2
        self.playerSprite.graphics = self.DefaultPlayer1
        self.oppSprite = game.p2 if p == "1" else game.p1
        self.oppSprite.graphics = self.DefaultPlayer2

        self.Update(game, p)

        self.GameStart()

        self.Battling(client)
        self.counter = 0

        self.Result()

        return

    def Battling(self, client):

        # Normal procedure
        while self.run:
            diff = self.DifficultySelection()

            if diff == -1:
                self.run = False
                return

            if not self.run:
                return

            self.threadGo = False
            questionObj = client.getQuestion(diff)
            self.threadGo = True

            if not self.run:
                return

            self.threadGo = True
```

```
            resultDict = self.DisplayQuestion(questionObj)

            if not self.run:
                return

            self.threadGo = False
            client.attack(diff, resultDict, questionObj)
            self.threadGo = True

            if not(resultDict["timeLeft"] and resultDict["result"]):
                self.Penalty()

            if not self.run:
                return

    def Result(self):

        self.counter += 1
        self.counter %= 10
        if self.counter == 0:
            self.Monitor(self.client, self.p)
        self.Update(self.client.getStatus(), self.p)
        self.displayUpdate = True
        self.DisplayTextBox()

        if self.state == "Won":
            self.Dialogue("Narrator", "The Battle has ended. ¥n Congratulations! You have won!")
        elif self.state == "Lost":
            self.Dialogue("Narrator", "The Battle has ended. ¥n You have lost. Try harder next time!
")
        elif self.state == "Disconnected":
            self.Dialogue("Narrator", "Your opponent has disconnected.")

    def LoopBundle(self):

        try:
            self.Monitor(self.client, self.p)
        except Exception as e:
            print(e)
            self.timeOutNo += 1
            if self.timeOutNo >= self.timeOutNo:
                raise e

        self.clock.tick(self.GameFPS)
        pygame.display.update()
        if not self.run:
            return True

    def Monitor(self, client, p):
        if self.threadGo:

            game = client.getStatus()
            tempState = game.GameState()
            self.Update(game, p)

            if tempState == "GAME":
                state = "Game"
            elif tempState == "1WIN":
                if p == "1":
                    state = "Won"
                elif p == "2":
                    state = "Lost"
                else:
                    state = "Disconnected"
            elif tempState == "2WIN":
                if p == "1":
                    state = "Lost"
                elif p == "2":
                    state = "Won"
                else:
                    state = "Disconnected"
```

```
            elif tempState == "DISCONNECTED":
                state = "Disconnected"
            else:
                state = "Disconnected"

        self.state = state
        if self.state != "Game":
            self.run = False

    def Update(self, game, p):

        if self.playerSprite.name == "Player 1" or self.oppSprite.name == "Player 2":
            self.playerSprite.name = game.p1.name if p == "1" else game.p2.name
            self.oppSprite.name = game.p2.name if p == "1" else game.p1.name

        if self.playerSprite.nowHP != game.p1HP if p == "1" or p == 1 else game.p2HP:
            self.displayUpdate = True
        elif self.oppSprite.nowHP != game.p2HP if p == "1" or p == 1 else game.p1HP:
            self.displayUpdate = True

        self.playerSprite.nowHP = game.p1HP if p == "1" or p == 1 else game.p2HP
        self.oppSprite.nowHP = game.p2HP if p == "1" or p == 1 else game.p1HP

        self.AttackStm = game.msg

    def DisplayTextBox(self):

        if self.displayUpdate:
            self.RefreshBattle()
            self.displayUpdate = False
        super().DisplayTextBox()
        self.DisplayLogBox()

    def PendingStart(self, menu):

        menu.Blit()

        for event in pygame.event.get():
            GlobalVar.Quit(event)
            counter = menu.Scrolling(event)
            if counter == 0:
                return True

        self.clock.tick(self.GameFPS)
        pygame.display.update()


    def DisplayLogBox(self):

        self.screen.blit(self.logBox, self.logPos)

        textSurf = self.logFont.render(self.AttackStm, 1, self.DefaultLogFontColor)

        if textSurf.get_width() >= self.logWidth - self.logSpacing * 2:
            textSurf = pyImage.ScaleGraphics(textSurf, self.logWidth - self.logSpacing * 2, True, Fa
lse)

        textRect = textSurf.get_rect()
        textRect.center = self.logCenter
        self.screen.blit(textSurf, textRect)


    def GameStart(self):

        startPrompt = pyText.TextRender(
            pyText.TextWrap(
                "The game is starting in:",
                self.BattleFont, int(self.width * 3 / 4)
            ),
            self.BattleFont, self.boxSpacing, "center", color=pygame.Color("darkred")
        )
```

```
        startRect = startPrompt.get_rect()
        startRect.top = self.displayHeight + self.boxShift
        startRect.centerx = int(self.width // 2)

        timer = pyTimer.Timer(self.DefaultStartTime, self.clock, int(self.textSize * 3), color=pyga
me.Color("red"))
        timerRect = timer.Surface().get_rect()
        timerRect.top = self.displayHeight + self.boxShift + startPrompt.get_height() + self.textSh
ift
        timerRect.centerx = int(self.width // 2)

        # Displays the pre-question prompt
        while timer.Update():

            if self.LoopBundle():
                break

            self.DisplayTextBox()
            self.screen.blit(startPrompt, startRect)
            self.screen.blit(timer.Surface(), timerRect)
            pygame.display.update()

            for event in pygame.event.get():
                GlobalVar.Quit(event)

    def Penalty(self):

        # Penalty
        startPrompt = pyText.TextRender(
            pyText.TextWrap(
                "Penalty (" + str(self.DefaultPenaltyTime) + " seconds)¥n Ending in:",
                self.BattleFont, int(self.width * 3 / 4)
            ),
            self.BattleFont, self.boxSpacing, "center", color=pygame.Color("darkred")
        )

        startRect = startPrompt.get_rect()
        startRect.top = self.displayHeight + self.boxShift
        startRect.centerx = int(self.width // 2)

        timer = pyTimer.Timer(self.DefaultPenaltyTime, self.clock, int(self.textSize * 3), color=py
game.Color("red"))
        timerRect = timer.Surface().get_rect()
        timerRect.top = self.displayHeight + self.boxShift + startPrompt.get_height() + self.textSh
ift
        timerRect.centerx = int(self.width // 2)

        # Displays the pre-question prompt
        while timer.Update():

            if self.LoopBundle():
                break

            self.DisplayTextBox()
            self.screen.blit(startPrompt, startRect)
            self.screen.blit(timer.Surface(), timerRect)
            pygame.display.update()

            for event in pygame.event.get():
                GlobalVar.Quit(event)

    def DisplayQuestion(self, questionObj):

        # Sequences in parallel
        def Display(stage):

            if stage >= 1:

                # Displays the background
                self.DisplayTextBox()
```

```
            # Displays the timer
            topCoord = self.displayHeight + self.boxShift
            timerSurf = timer.Surface()
            self.screen.blit(timerSurf, (self.width - self.boxShift * 2 - timerWidth, topCoord))

            # Display the timeLeft message
            topCoord += int((int(self.textSize * 1.5) - self.textSize) / 2)
            self.screen.blit(timeLeftSurf,
                             (self.width - self.boxShift * 2 - timerWidth - self.textShift -
                              timeLeftSurf.get_width(), topCoord))

            # Displays the question label
            topCoord += int((int(self.textSize * 1.5) - self.textSize) / 2)
            self.screen.blit(questionLabel, (self.boxShift, topCoord))

            # Displays the question
            topCoord += self.boxSpacing + self.textSize
            self.screen.blit(questionSurf, (self.boxShift, topCoord))

            # Displays the input box
            topCoord += questionHeight + self.boxSpacing
            self.screen.blit(answerPromptSurf, (self.boxShift, topCoord))
            self.screen.blit(questionInput.Surface(), (self.boxShift + answerPromptSurfWidth + s
elf.textShift, topCoord))

            if stage >= 2:

                # Displays the results
                topCoord += answerPromptSurfHeight + self.boxSpacing
                self.screen.blit(endSurf, (self.boxShift, topCoord))

        self.LoopBundle()

        pass

    questionObj = mDebug.EasyMode(questionObj)
    question = questionObj["question"]
    answer = questionObj["answer"]
    time = int(questionObj["time"])

    # Readies the player to answer the question
    startPrompt = pyText.TextRender(
        pyText.TextWrap(
            "You are about to answer a question.¥n" + "Time limit : " + str(time)
            + "¥n The question will be shown in:",
            self.BattleFont, int(self.width * 3 / 4)
        ),
        self.BattleFont, self.boxSpacing, "center", color=pygame.Color("darkred")
    )

    startRect = startPrompt.get_rect()
    startRect.top = self.displayHeight + self.boxShift
    startRect.centerx = int(self.width // 2)

    timerA = pyTimer.Timer(self.DefaultWaitTime, self.clock, int(self.textSize * 3), color=pyga
me.Color("red"))
    timerRect = timerA.Surface().get_rect()
    timerRect.top = self.displayHeight + self.boxShift + startPrompt.get_height() + self.textSh
ift
    timerRect.centerx = int(self.width // 2)

    # Displays the pre-question prompt
    while timerA.Update():

        if self.LoopBundle():
            break

        self.DisplayTextBox()
        self.screen.blit(startPrompt, startRect)
        self.screen.blit(timerA.Surface(), timerRect)
```

```
            pygame.display.update()

            for event in pygame.event.get():
                GlobalVar.Quit(event)

        del timerA

        # Renders the question text
        questionLabel = self.BattleFont.render("Question", 1, pygame.Color("black"))
        questionText = pyText.TextWrap(mDebug.QuestionType(questionObj["comment"]) + question,
                                       self.MathFont, self.width - 2 * self.boxShift)
        questionSurf = pyText.TextRender(questionText, self.MathFont, int(self.boxSpacing//3))
        questionHeight = questionSurf.get_height()

        # Creates answer prompt
        answerPromptSurf = self.BattleFont.render("Input your answer: ", 1, pygame.Color("black"))
        answerPromptSurfWidth = answerPromptSurf.get_width()
        answerPromptSurfHeight = answerPromptSurf.get_height()
        questionInput = pyInput.TextInput(self.MathInputFont,
                                          self.width - 2 * self.boxShift -
                                          answerPromptSurfWidth - self.textShift, self.clock)

        # Creates a timer
        timer = pyTimer.Timer(time, self.clock, int(self.textSize * 1.5), color=pygame.Color("red
"))
        timeLeftSurf = self.BattleFont.render("Time Left:", 1, pygame.Color("black"))
        timerWidth = pygame.font.Font(GlobalVar.BlockyFont, int(self.textSize * 1.5)).size("00:00.0
")[0]

        # Stage 1: Question Sequence
        while timer.Update() and not questionInput.Typing():
            if self.LoopBundle():
                break
            Display(1)

        # Assigning results
        timeLeft = timer.Time()
        userInput = questionInput.Text()

        # If-else states to determine the end-game message
        if timeLeft == 0:
            result = False
            endMessage = "Time's up!"
        else:
            result = mQuestions.AnswerChecker(answer, userInput)
            if result:
                endMessage = "You are correct!"
            else:
                endMessage = "You are incorrect!"

        endMessage += " Correct Answer: " + (f"{answer:.3g}" if type(answer) == type(1.1) else str(a
nswer)) ¥
                      + "    ... (Wait for " + str(self.DefaultPastTime) + " seconds)"
        endSurf = self.BattleFont.render(endMessage, 1, pygame.Color("dodgerblue4"))

        timerB = pyTimer.BgTimer(self.DefaultPastTime, self.clock)

        # Stage 2: Displaying results
        while timerB.Update():

            if self.LoopBundle():
                break
            Display(2)

            for event in pygame.event.get():
                GlobalVar.Quit(event)

        mDebug.SaveState(questionObj, userInput, result, timeLeft)

        return {
            "result" : result,
```

```
    "timeLeft" : timeLeft
        }
```

22. mMultiplayerClient.py

```
"""
mMultiplayerClient.py
"""

import json
import socket
import pickle
import time
import struct

import mMultiplayerObjects
from mMultiplayerPackets import *

DEBUG = False

def Debug(*args):
    if DEBUG:
        print(*args)

class Client:

    def __init__(self, IP, port, name):
        self.timeOutMax = 5
        self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.playerNo = self.connect(IP, port, name)
        self.client.settimeout(10)
        self.timeOutNo = 0
        self.tempGame = None

    def connect(self, IP, port, name):

        # Problem: in 'Open Server',
        # The server can start, the client can connect, but
        # the client crashes

        # Hypothesis:
        # The server is not prepared to accept connections when
        # the client is connected

        # Solutions:
        # - Place server binding as close to connection as possible
        # - Use exception handling to deal with disconnections

        LargeRetryNo = 0

        connected = False
        while not connected:

            try:

                print("Connecting to: " + IP + ":" + str(port))

                run = True
                RetryNo = 0
                while run:
                    try:
                        self.client.connect_ex((IP, port))
                    except Exception as e:
                        RetryNo += 1
                        if RetryNo > 10:
                            raise e
                    else:
                        run = False
                        break

                Debug("Handshaking: sening name.")
                self.client.send(str.encode(Send(handshaking, name)))
                Debug("Handshaking: sent name.")
                Debug("Handshaking: receiving data.")
```

```
            recvData = self.client.recv(4096)
            Debug("Handshaking: received data.")
            List, recvData = Receive(recvData.decode())
            data = List[0][DATA]
            Debug("Handshaking: returning data: ", data)

            return data

        except Exception as e:
            print(e)
            raise e

def getQuestion(self, diff):
    time.sleep(0.3)

    sendDict = Send(question, diff)

    self.client.send(str.encode(sendDict))

    recvData = self.client.recv(4096)
    List, recvData = Receive(recvData.decode())
    if List[0][TYPE] == question:
        data = List[0][DATA]
        return data
    else:
        self.getQuestion(diff)

def getStatus(self):

    for i in range(self.timeOutMax):
        try:
            sendDict = Send(get)

            Debug("Get: sending request: ", sendDict)
            self.client.send(str.encode(sendDict))
            Debug("Get: sent request.")

            Debug("Get: receiving data.")
            recvData = self.client.recv(4096)
            Debug("Get: received data.")
            List, recvData = Receive(recvData.decode())
            Debug("Get: unpacking: ", List, recvData)
            if List[0][TYPE] == get:
                data = List[0][DATA]
                Debug("Get: returning data: ", data)
                return data
            else:
                self.getStatus()
        except Exception as e:
            print("Get: exception: ", e)
            raise e
            time.sleep(0.5)

def attack(self, diff, resultDict, questionObj):

    reply = mMultiplayerObjects.ResultObj(diff, resultDict, questionObj)
    sendDict = Send(attack, reply)
    self.client.send(str.encode(sendDict))
```

23. mMultiplayerObjects.py

```
"""
mMultiplayerObject.py
"""


import mSprite
import pygame
import random
import GlobalVar


class ResultObj:

    __slots__ = "diff", "resultDict", "questionDict"

    def __init__(self, diff, resultDict, questionDict):
        self.diff = diff
        self.resultDict = resultDict
        self.questionDict = questionDict


class Game:

    PlayerHP = GlobalVar.PlayerHP
    PlayerAP = GlobalVar.AttackDmg
    RandomRange = 0.05
    TimerRange = 1 / 3

    def __init__(self):

        self.p1 = mSprite.Sprite("Player 1", self.PlayerHP, None)
        self.p2 = mSprite.Sprite("Player 2", self.PlayerHP, None)
        self.p3 = 0

        self.p1HP = self.PlayerHP
        self.p2HP = self.PlayerHP

        self.p1Ready = False
        self.p2Ready = False

        self.p1Alive = True
        self.p2Alive = True

        self.msg = ""
        self.state = "WAITING"
        # WAITING: waiting for another player to join
        # GAMESTART: the game has started
        # GAME: the game is in progress
        # 1WIN: player 1 has won
        # 2WIN: player 2 has won
        # DISCONNECTED: the other player has disconnected

    def Init(self, p):

        if self.p3 == 0:
            self.p3 = p
        elif self.p3 == 1:
            if p == 2:
                self.p1 = None
                self.p2 = None
        elif self.p3 == 2:
            if p == 1:
                self.p1 = None
                self.p2 = None

    def Attendance(self, p):

        if p == 1:
```

```
            self.p1Ready = True
        elif p == 2:
            self.p2Ready = True

    def Disconnected(self, p):
        if self.state == "GAME":
            self.state = "DISCONNECTED"
        else:
            if p == 1:
                self.p1Ready = False
            elif p == 2:
                self.p2Ready = False

    def Update(self):
        if self.state == "GAME":
            if self.p1Alive and self.p2Alive:
                return
            elif self.p1Alive and not self.p2Alive:
                self.state = "1WIN"
            elif not self.p1Alive and self.p2Alive:
                self.state = "2WIN"
        elif self.p1Ready and self.p2Ready and self.state == "WAITING":
            self.state = "GAME"

    def DealDamage(self, player, resultObj):

        diff = resultObj.diff
        resultDict = resultObj.resultDict
        questionObj = resultObj.questionDict
        AP = self.CalculateAP(diff, resultDict, questionObj)
        if AP:
            self.msg = (self.p1.name if player == 1 else self.p2.name) + " has dealt " ¥
                    + str(AP) + " damage on " + (self.p1.name if player == 2 else self.p2.name) + ".
"
            if player == 1:
                self.p2HP -= AP
                if self.p2HP <= 0:
                    self.p2HP = 0
                    self.p2Alive = False
            elif player == 2:
                self.p1HP -= AP
                if self.p1HP <= 0:
                    self.p1HP = 0
                    self.p1Alive = False

    def CalculateAP(self, diff, resultDict, questionDict):

        result = resultDict["result"]
        timeLeft = resultDict["timeLeft"]
        diffPt = questionDict["diffPt"]
        timeLimit = questionDict["time"]

        if result and timeLeft:
            dmg = self.PlayerAP[diff]
            dmg *= 1 - self.RandomRange / 2 + self.RandomRange * random.random()
            dmg *= (1 - self.TimerRange) + self.TimerRange * timeLeft / timeLimit
            dmg *= 1 + diffPt
            dmg *= 1 + diffPt
            return int(dmg)
        else:
            return 0

    def GameState(self):
        return self.state
```

24. mMultiplayerPackets.py

```
"""
mMultiplayerPackets.py
"""


"""
Packet Format:
type
data
"""

import json
import pickle
import base64

handshaking = "HANDSHAKING"
get = "GET"
question = "QUESTION"
attack = "ATTACK"

TYPE = 'type'
DATA = 'data'

def Send(type, data = None):

    sendData = str(base64.b32encode(pickle.dumps(data)))
    sendDict = {TYPE: type,
                DATA: sendData}

    return json.dumps(sendDict)


def Receive(repos):

    counter = 0
    inside = False
    insideString = False
    stringCounter = 0
    returnList = []
    tempIndex = 0

    for index, character in enumerate(repos):

        if character == "{":
            counter += 1
            inside = True
        if character == "}":
            counter -= 1
            if inside and counter == 0:
                inside = False

                jsonDict = json.loads(repos[tempIndex:index+1])

                jsonDict[DATA]= jsonDict[DATA].strip("¥'b")
                loadData = bytes(base64.b32decode(jsonDict[DATA]))
                jsonDict[DATA] = pickle.loads(loadData)

                returnList.append(jsonDict)
                tempIndex = index+1

    return returnList, repos[tempIndex:]
```

25. mMultiplayerServer.py

```
"""
mMultiplayerServer.pygame
"""

import json
import socket
import struct
import threading
import mSprite
import pygame
import random

import mQuestions
import mQuestionRead
import pickle
import mMultiplayerObjects
import re
from mMultiplayerPackets import *
import GlobalVar

DEBUG = False

def Debug(*args):
    if DEBUG:
        message(*args)

def message(*args):
    print("Server: ", *args)

QuestionDict = mQuestionRead.ReadQuestion("files/Questions.json")

resendMax = 5


timeAllowance = 15
def GetQuestion(diff):

    questionObj = mQuestions.QuestionGenerator(diff, QuestionDict)
    questionObj["time"] = int((questionObj["time"]+timeAllowance)//2)
    return questionObj


def ThreadClient(player, conn, addr, gameID):

    resendCounter = 0

    try:

        conn.settimeout(1000)

        handShaking = True
        repos = ""

        run = True
        while run:

            try:

                # Receives new messages
                newStr = conn.recv(4096).decode()
                Debug("Received string:", newStr)

                if newStr:

                    # Appends it into the program buffer
                    repos += newStr
                    Debug("Repository (before):", repos)

                    # Isolate requests from the buffer
```

```
            processList, repos = Receive(repos)
            Debug("Repository (after):", repos)
            Debug("Process List:", processList)

            for i in processList:

                Debug("Received Request:", i)
                Debug("Request type:", i[TYPE])

                if handShaking:

                    if i[TYPE] == handshaking:

                        gameList[gameID].Attendance(player)

                        data = Send(handshaking, str(player))
                        conn.send(str.encode(data))

                        name = i[DATA]
                        message("[Name] " + name)
                        message("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=" + "¥n")

                        if player == 1:
                            gameList[gameID].p1.name = name
                        elif player == 2:
                            gameList[gameID].p2.name = name

                        handShaking = False

                    else:

                        if i[TYPE] == get:

                            data = gameList[gameID]
                            sendData = str.encode(Send(get, data))
                            conn.send(sendData)

                        elif i[TYPE] == question:

                            run2 = True
                            while run2:
                                try:
                                    diff = i[DATA]
                                    data = GetQuestion(diff)
                                    sendData = str.encode(Send(question, data))
                                    conn.send(sendData)
                                except:
                                    pass
                                else:
                                    run2 = False

                        elif i[TYPE] == attack:

                            resultDict = i[DATA]
                            gameList[gameID].DealDamage(player, resultDict)

                gameList[gameID].Update()

            else:
                run = False
                break

        except socket.timeout as e:
            message(e)
            resendCounter += 1
            if resendCounter >= resendMax:
                message("Server:", e)
                run = False
                break
        else:
            resendCounter = 0
```

```
    except socket.timeout as e:
        message(e)

    finally:
        message(str(addr) + " has lost connection.")
        gameList[gameID].Disconnected(player)
        if gameList[gameID].GameState() == "WAITING":
            global playerNo
            playerNo -= 1
        conn.close()


def mainfunc(ip="", port=""):

    Host = ip
    if port:
        Port = port
    else:
        Port = GlobalVar.DefaultPort


    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM, socket.IPPROTO_IP)

    global gameList
    gameList = {}

    global playerNo
    playerNo = 0

    def ResourceManagement():

        for g in gameList.keys():
            if gameList[g].GameState() in ["DISCONNECTED", "1WIN", "2WIN"]:
                del g

    gameProcessingThread = threading.Thread(target=ResourceManagement)
    gameProcessingThread.start()

    try:
        server.bind((Host, Port))
    except:
        server.bind((Host, GlobalVar.DefaultPort))


    server.listen(2)
    message("Up n' runnin'!")

    if ip:
        message("Connect to this server by: ", str(ip) + ":" + str(Port))
    else:
        message("Connect to this server by: ", socket.gethostbyname(socket.gethostname()) + ":" + s
tr(Port))
    print()

    while True:

        conn, addr = server.accept()
        message("=-=-=-=-= New Connection =-=-=-=-=")
        message("[IP]", addr)

        gameNo = playerNo // 2
        playerNo += 1
        if playerNo % 2 == 1:
            gameList[gameNo] = mMultiplayerObjects.Game()
        gamePlayerNo = 2 if playerNo % 2 == 0 else 1


        message("[Game] " + str(gameNo))
        message("[Side] " + str(gamePlayerNo))
        message("[Player] " + str(playerNo))
```

```
        playerThread = threading.Thread(target=ThreadClient, args=(gamePlayerNo, conn, addr, gameN
o))
        playerThread.start()


if __name__ == '__main__':

    ip = input("Input IP (skip for defaults):¥n>>> ")
    port = input("Input port number (skip for defaults):¥n>>> ")
    mainfunc()
```

26. mQuestionErrors.py

```
"""
mQuestionErrors.py
"""

class Error(Exception):

    def __init__(self, message):
        self.message = message

class FractionError(Error):

    def __init__(self, message = "Fraction too large. Regeneration is required."):
        self.message = message

class RangeError(Error):

    def __init__(self, message):
        self.message = message

class ArrayError(Error):

    def __init__(self):
        self.message = "Attribute range not wide enough to produce specified number of objects."

class TypeError(Error):

    def __init__(self, message):
        self.message = message

class ValidationError(Error):

    def __init__(self, message = "The generated variables have failed validation. Regeneration is r
equried."):
        self.message = message
```

27. mQuestionFunctions.py

```
"""
mQuestionFunctions.py
"""

import random
import math
from fractions import *
from decimal import *
from mQuestionErrors import *

def FracValue(valLower, valUpper, numLower, numUpper):

    n, d, v = RandFraction(valLower, valUpper, numLower, numUpper)
    return v

def RandFraction(valLower, valUpper, numLower, numUpper):

    run = True
    while run:
        try:
            value = Decimal(RandomFloat(valLower, valUpper, 15))
            denominator = abs(RandomInt(numLower, numUpper))
            numerator = int(denominator * value)
            value = Fraction(numerator / denominator)
            hcf = HCF(denominator, numerator)
            denominator //= hcf
            numerator //= hcf
        except:
            pass
        else:
            if value:
                run = False

    return numerator, denominator, value

def RandomFloat(lower, upper, precision=3):

    num = math.ceil((lower + (upper - lower) * random.random()) * 10**precision) / 10**precision

    if num:
        return num
    else:
        return RandomFloat(lower, upper, precision)

def HCF(a, b):

    a = int(a)
    b = int(b)

    if a > b:
        a, b = b, a
    a = abs(a)
    b = abs(b)

    if a < 0 or b < 0:
        raise RangeError("a, b must be greater than 0!")

    rem = b % a
    if rem == 0:
        return a
    else:
        return HCF(rem, a)


def RandomInt(lower, upper, elower=0, eupper=0):

    if upper < lower:
        lower, upper = upper, lower
```

```
    if 0 in [lower,upper]:
        return random.randint(lower, upper)
    else:
        i = random.randint(lower, upper)
        while i == 0 or elower <= i <= eupper:
            i = random.randint(lower, upper)

    return i

def FractionRepresentation(n, d):

    f = Fraction(n/d).limit_denominator()
    n, d = f.numerator, f.denominator
    return str(n) + (("/" + str(d)) if d != 1 else "")
```

28. mQuestionObjects.py

```
"""
mQuestionObjects.py
"""


import random
import math
import re
from fractions import *
from decimal import *
from mQuestionFunctions import *
from mQuestionErrors import *
import GlobalVar

DISPLAYUNICODE = GlobalVar.SettingDict["Unicode"]

switch = {
        "randint" : "RANDINT",
        "fraction" : "FRACTION",
        "randfloat" : "RANDFLOAT",
        "as" : "AS",
        "array" : "ARRAY",
        "eval" : "EVAL",
        "multias" : "MULTIAS",
        "gs" : "GS",
        "quadratic" : "QUADRATIC",
        "polynomial" : "POLYNOMIAL"
    }

def StrToBool(string):
    if isinstance(string, str):
        if string.lower() == "true":
            return True
        else:
            return False
    else:
        return string

def ArgsString(args):

    if len(args) == 1:
        argsString = "(" + str(args[0]) + ")"
    else:
        argsString = str(tuple(args))
    return argsString


def ExecuteFunction(obj, function, args):
    if function in ["eval", "evalRandStr", "validate"]:
        args = ",".join(args)
        return getattr(obj, function)(args)
    else:
        return getattr(obj, function)(*args)

class EVAL:

    def __init__(self):
        pass

    def eval(self, string):
        return eval(string)

    def evalInt(self, string, max = 10000000):
        inta = eval(string)
        max = int(max)
        if abs(inta) > max:
            raise RangeError("Fraction too large. Regeneration is required.")
        return inta
```

```
    def evalFloat(self, string):
        return eval(string)

    def evalFraction(self, string, vmax = 100000, dmax = 50000):

        try:
            f = Fraction(eval(string)).limit_denominator()
        except:
            f = Fraction(string).limit_denominator()

        n = f.numerator
        d = f.denominator
        vmax = int(vmax)
        dmax = int(dmax)
        if (abs(d) > dmax and n != 1) or abs(n) > dmax or abs(f) > vmax:
            raise FractionError
        if d == 1:
            return str(f.numerator)
        else:
            return str(f.numerator) + "/" + str(f.denominator)

    def evalRand(self, lower, upper, elower = 0, eupper = 0):
        return RandomInt(int(lower), int(upper), int(elower), int(eupper))

    def evalRandStr(self, strList):
        strList = strList.strip("()")
        return re.split(",", strList)[random.randrange(len(re.split(",", strList)))]

    def validate(self, condition):
        try:
            if eval(condition):
                return ""
            else:
                raise ValidationError
        except:
            raise ValidationError

    def varDisplay(self, coeff, exponent = '1', sign = False, space = False, varName = "x", vmax = 1
00000, dmax = 50000):

        self.evalFraction(coeff, vmax = vmax, dmax = dmax)

        sign = StrToBool(sign)
        space = StrToBool(space)

        showSign = sign
        showSpace = space
        n = int(exponent)
        if DISPLAYUNICODE:
            unicodeDict = {
                '0' : '¥u2070',
                '1' : '¥u2071',
                '2' : '¥u00B2',
                '3' : '¥u00B3',
                '4' : '¥u2074',
                '5' : '¥u2075',
                '6' : '¥u2076',
                '7' : '¥u2077',
                '8' : '¥u2078',
                '9' : '¥u2079'
            }
            displayN = ""
            for i in exponent:
                displayN += unicodeDict[i]
        else:
            displayN = '^' + exponent


        coeff = Fraction(self.evalFraction(coeff))

        if len(varName) > 1:
```

```
            varName = "(" + varName + ")"

        varName = (varName + (displayN if n != 1 else "") if n else "")
        f = Fraction(coeff).limit_denominator()
        numCoeff = str(abs(f.numerator)) if (abs(f.numerator) != 1 or n == 0) else ""
        denCoeff = ("/" + str(abs(f.denominator))) if f.denominator != 1 else ""
        space = ""

        if float(f) != 0:
            if int(coeff) == coeff:
                if showSpace:
                    return ("- " if coeff < 0 else ("+ " if showSign else "")) + numCoeff + varName + s
pace
                else:
                    return ("-" if coeff < 0 else ("+" if showSign else "")) + numCoeff + varName + spa
ce
            else:
                if showSpace:
                    return ("- " if coeff < 0 else ("+ " if showSign else "")) + numCoeff + varName + d
enCoeff + space
                else:
                    return ("-" if coeff < 0 else ("+" if showSign else "")) + numCoeff + varName + den
Coeff + space
        else:
            return ""



class ARRAY:

    # Add "" when specifying objType
    # No need to add "" when specifying attr

    def __init__(self, objType, num, args):

        self.array = []
        valArray = []

        if objType in ["randint", "randfloat", "fraction"]:

            for i in range(num):
                fin = False
                count = 0
                while not fin:
                    tempObj = eval(switch[objType] + ArgsString(args))
                    if tempObj.value not in valArray:
                        self.array.append(tempObj)
                        valArray.append(tempObj.value)
                        fin = True
                    else:
                        count += 1
                        if count > 50:
                            raise ArrayError

            self.sortedArray = sorted(self.array, key=lambda x: x.value, reverse=False)
        else:
            raise RangeError

    def sortedAttr(self, n, attr):

        n = int(n)

        return getattr(self.sortedArray[n], attr)

    def attr(self, n, attr):

        n = int(n)

        return getattr(self.array[n], attr)

    def func(self, n, func, args):
```

```
        n = int(n)

        return ExecuteFunction(self.array[n], func, args)


class RANDINT:

    def __init__(self, lower, upper, elower=0, eupper=0):

        self.lower = lower
        self.upper = upper

        self.value = RandomInt(lower, upper, elower, eupper)
        self.sign = self.sign = "positive" if self.value > 0 else "negative"
        if self.value > 0:
            self.ord = self.ordValue()

    def ordValue(self):

        if self.value // 10 % 10 != 1:
            if self.value % 10 == 1:
                suffix = "st"
            elif self.value % 10 == 2:
                suffix = "nd"
            elif self.value % 10 == 3:
                suffix = "rd"
            else:
                suffix = "th"
        else:
            suffix = "th"

        return str(self.value)+suffix


class RANDFLOAT:

    __slots__ = "lower", "upper", "precision", "value"
    def __init__(self, lower, upper, precision=3):

        self.lower = lower
        self.upper = upper
        self.precision = precision

        self.value = RandomFloat(self.lower, self.upper, self.precision)




class FRACTION:

    def __init__(self, valLower, valUpper, numLower, numUpper):

        self.valLower = valLower
        self.valUpper = valUpper
        self.numLower = numLower
        self.numUpper = numUpper

        self.numerator, self.denominator, self.value = RandFraction(self.valLower, self.valUpper, s
elf.numLower, self.numUpper)

class AS:

    def __init__(self, a1, a2, d1, d2):
        self.first = RandomInt(a1, a2)
        self.diff = RandomInt(d1, d2)
        self.sign = "positive" if self.diff > 0 else "negative"
        self.inequality = "greatest" if self.diff > 0 else "least"
        self.limit = "less than" if self.diff > 0 else "greater than"
```

```
    def valueByNo(self, args):
        num = int(args)
        return self.first + (num - 1) * self.diff

    def sumByNo(self, args):
        num = int(args)
        return int(num/2 * (self.first + self.valueByNo(num)))

    def product(self, lower, upper):
        pro = 1
        for n in range(int(lower), int(upper) + 1):
            pro *= self.valueByNo(n)
        return pro


class MULTIAS(AS):

    def __init__(self, lower, upper):
        super().__init__(10, 20, 30, 40)
        self.base = RandomInt(int(lower), int(upper))
        self.first = 0
        self.diff = self.base
        self.sign = "positive" if self.diff > 0 else "negative"


class GS:

    def __init__(self, gsType = None, spec = None):

        if spec:
            num = 2
        else:
            if gsType:
                if gsType.lower() == "int":
                    num = 0
                elif gsType.lower() == "dec":
                    num = 1
                else:
                    num = random.randint(0, 1)
            else:
                num = random.randint(0, 1)

        if num == 0:
            r = RandomInt(-5, 5, -1, 1)
            a = Decimal(RandomInt(-15, 15)) / Decimal((r ** RandomInt(1, 4)))

        elif num == 1:
            run = True
            while run:
                n, d, v = RandFraction(-1, 1, 2, 7)
                r = Decimal(n) / Decimal(d)
                if r != Decimal(1) and r != Decimal(-1):
                    run = False

            a = Decimal(RandomInt(-5, 5)) / Decimal((r ** RandomInt(1, 4)))

        else:
            a = Decimal(spec[0])
            r = Decimal(spec[1])


        self.first = a
        self.ratio = r
        self.sign = "positive" if self.ratio > 0 else "negative"

    def valueByNo(self, n):
        n = Decimal(n)
        return self.first * (self.ratio ** (n - 1))

    def sumByNo(self, n):
```

```
        if n == "inf":
            return self.first / (1 - self.ratio)
        else:
            n = Decimal(n)
            return self.first * (self.ratio ** n - 1)/(self.ratio - 1)

    def sumOdd(self, n):

        newObj = GS(spec=(self.first, self.ratio ** 2))
        return newObj.sumByNo(n)



class POLYNOMIAL:

    def __init__(self, deg, leadingCoeff = 0, fractionSpec = (-2, 2, 5, -5), intSpec = (-7, 7), genM
ode = None, roots = None):

        self.roots = []
        self.numerator = []
        self.denominator = []
        self.deg = deg

        if not fractionSpec:
            fractionSpec = (-2, 2, 5, -5)

        if not intSpec:
            intSpec = (-7, 7)

        if roots:
            try:
                self.roots = [Fraction(root) for root in roots]
            except:
                self.roots = (roots, )
            for i in self.roots:
                self.denominator.append(i.denominator)
                self.numerator.append(i.numerator)
        else:
            if genMode:
                if genMode == "int":
                    intProb = 1
                elif genMode == "frac":
                    intProb = 0
                else:
                    intProb = 0.5
            else:
                intProb = 0.5

            for i in range(deg):

                if random.random() > intProb:
                    num = 1
                else:
                    num = 0

                if num == 1:
                    n, d, v = RandFraction(*fractionSpec)
                    self.roots.append(v)
                    self.numerator.append(n)
                    self.denominator.append(d)
                elif num == 0:
                    v = RandomInt(*intSpec)
                    self.roots.append(v)
                    self.numerator.append(v)
                    self.denominator.append(1)
                else:
                    raise RangeError("Implementation Error.")

        self.ansRoots = [FractionRepresentation(root.numerator, root.denominator) for root in self.
roots]
```

```
        newList = [1]

        for i in range(deg):
            oldList = newList
            newList = []
            coeffMax = i + 1
            newRoot = self.roots[i]
            for j in range(0, coeffMax+1):
                if j == 0:
                    newList.append(-1 * newRoot * oldList[j])
                elif j == coeffMax:
                    newList.append(oldList[j - 1])
                else:
                    newList.append(-1 * newRoot * oldList[j] + oldList[j - 1])

        self.coeffList = newList
        if leadingCoeff:
            self.leadingCoeff = leadingCoeff
        else:
            self.leadingCoeff = 1
            for i in self.denominator:
                self.leadingCoeff *= i
        self.displayCoeffList = [Fraction((self.leadingCoeff * coeff)).limit_denominator() for coef
f in self.coeffList]


    def coeff(self, n):
        n = int(n)
        return self.coeffList[n]

    def displayCoeff(self, n):
        n = int(n)
        return self.displayCoeffList[n]

    def value(self, x):
        sum = 0
        try:
            x = Decimal(float(x))
        except:
            x = Decimal(float(eval(x)))
        for n in range(self.deg + 1):
            sum += Decimal(float(self.displayCoeffList[n])) * (x ** n)
        return Fraction(sum).limit_denominator()

    def root(self, n):
        n = int(n)
        return self.ansRoots[n]

    def denon(self, n):
        n = int(n)
        return self.denominator[n]

    def numer(self, n):
        n = int(n)
        return self.numerator[n]


class QUADRATIC(POLYNOMIAL):

    def __init__(self, leadingCoeff = 0, genMode = None, fractionSpec = (-2, 2, -5, 5), intSpec = (-
7, 7), roots = None):

        super().__init__(2, leadingCoeff, fractionSpec, intSpec, genMode, roots)
        self.rootSum = self.coeffList[1]
        self.rootProduct = self.coeffList[0]
        self.symmetryAxis = - self.displayCoeffList[1]/2/self.displayCoeffList[2]
        self.vertex = self.displayCoeffList[0] - (self.displayCoeffList[1] ** 2/4/self.displayCoeff
List[2])
        self.discriminant = (self.displayCoeffList[1]) ** 2 - 4 * self.displayCoeffList[0] * self.d
isplayCoeffList[2]
        self.limit = "maximum" if self.displayCoeffList[2] < 0 else "minimum"
```

```
if __name__ == '__main__':

    k = ARRAY("randint", 4, (0, 3))
    for i in range(4):
        print(k.array[i].value)
```

29. mQuestionRead.py

```
"""
mQuestionRead.py
"""

import json
import mErrors


def DisplayDictList(d):
    for i in d.keys():
        print(str(i) + ' : ')
        for num, j in enumerate(d[i]):
            print(f" {str(num)+'.':<4}", end="")
            print(j)


def ObjectJSONGenerator():

    print("Object (input -1 to end):")
    obj = {}
    readObj = True
    while readObj:
        objName = input(f"{'Object Name:':30} ")
        if objName == "-1":
            runObj = False
            break
        objType = input(f"{'Object Type (objname):':30} ")
        args = input(f"{'Arguments (args):':30} ")
        obj[objName] = {
            "objname": objType,
            "args": args
        }

        code = json.dumps(obj)

    return code


def WriteQuestion(filename):

    try:
        questionFile = open(filename, 'r')
        questionList = json.loads(questionFile.read())
    except:
        questionList = []

    run = True
    while run:

        print("Please input a question: (Insert comment = -1 to end)")
        comment = input(f"{'Comment:':30} ")
        if comment == '-1':
            run = False
            break
        diff = int(input(f"{'Difficulty (diff):':30} "))
        diffpt = float(input(f"{'Difficulty Variation (diffpt):':30} "))
        time = int(input(f"{'Answering Time (time):':30} "))

        obj = input(f"{'Object (obj)':30} ")
        questionStm = input(f"{'Question (questionStm):':30} ")
        answerStm = input(f"{'Answer (answerStm):':<30} ")

        questionList.append({
            "comment": comment,
            "diff": diff,
            "diffPt": diffpt,
            "time": time,
            "obj": obj,
            "questionStm": questionStm,
```

```
            "answerStm": answerStm
        })

    storeStr = json.dumps(questionList, indent=4)

    questionFile = open(filename, 'w')
    questionFile.write(storeStr)


def ReadQuestion(filename):

    def RemoveKey(dictionary, key):
        r = dict(dictionary)
        del(r[key])
        return r

    try:
        questionFile = open(filename, 'r')
        questionList = json.loads(questionFile.read())
    except Exception as e:
        raise mErrors.QuestionError
    else:
        QuestionDict = {}

        for i in range(1,7):
            QuestionDict[i] = []

        try:
            for q in questionList:
                diff = q["diff"]
                q = RemoveKey(q, "diff")
                q["obj"] = json.loads(q["obj"])
                QuestionDict[diff].append(q)
        except Exception as e:
            print(q["comment"], "an error has occured")
            raise e

        return QuestionDict
```

30. mQuestions.py

```
"""
mQuestions.py
"""


'''
Question Formats: Dictionary
diff : Difficulty of question
diffpt : Difficulty variation
time : answer time
coeffNo : no of coefficients <- Probably not
coeffOpt : options for coefficients <- Keep?
answer : string
question : string
'''

import re
import random
from math import *
from mQuestionRead import * # for testing in main()
from mQuestionObjects import *
import mDebug
import mQuestionErrors

randIntList = [i for i in range(-7, 0)] + [i for i in range(1, 8)] + [FracValue(-2, 2, -5, 5) for i
in range(7)]
randList = list(
    set([i for i in range(-7, 0)] + [i for i in range(1, 8)] + [FracValue(-2, 2, -5, 5) for i in rang
e(7)]))
random.shuffle(randIntList)
random.shuffle(randList)


def Redefine():

    global randIntList
    global randList
    randIntList = [i for i in range(-7, 0)] + [i for i in range(1, 8)] + [Fraction(FracValue(-2, 2,
-5, 5)).limit_denominator() for i in range(7)]
    randList = list(
        set([i for i in range(-7, 0)] + [i for i in range(1, 8)] + [Fraction(FracValue(-2, 2, -5, 5)).
limit_denominator()  for i in range(7)]))


def Shuffle():

    random.shuffle(randIntList)
    random.shuffle(randList)
    Redefine()
    random.shuffle(randIntList)
    random.shuffle(randList)


class Ques:

    def __init__(self):
        self.last2 = None
        self.last = None


Q = Ques()


def QuestionGenerator(diff, questionDict, index = -1):

    for i in range(5):
        try:

            run = True
```

```
            while run:
                questionObj = QuestionFetch(diff, questionDict, index)
                if questionObj["comment"] not in [Q.last, Q.last2]:
                    run = False
                    Q.last = Q.last2
                    Q.last2 = questionObj["comment"]

            comment = questionObj["comment"]
            diffPt = questionObj["diffPt"]
            time = questionObj["time"]
            questionStm = questionObj["questionStm"]
            answerStm = questionObj["answerStm"]
            obj = questionObj["obj"]

            question, answer = QuestionFiller(obj, questionStm, answerStm)
            return {
                "comment": comment,
                "diffPt": diffPt,
                "time": time,
                "question": question,
                "answer": answer
            }
        except Exception as e:
            print("Exception: ", e)
            try:
                print("Problem: ¥"", comment, "¥"")
            except:
                pass

    raise mQuestionErrors.Error("Problem occured in " + "Problem: ¥"" + comment + "¥"")


def QuestionFetch(diff, questionDict, index = -1):

    try:
        if index != -1:
            return questionDict[diff][int(index)]
        else:
            if diff == 7:
                fetchDiff = random.randint(1, 6)
            elif diff in [1, 2, 3]:
                fetchDiff = diff if random.random() > 0.2 else (diff + 2) % 6 + 1
            elif diff in [4, 5, 6]:
                if diff == 4:
                    newDiff = 4
                elif diff == 5:
                    newDiff = 4 if random.random() > 0.7 else 5
                elif diff == 6:
                    newDiff = 5 if random.random() > 0.7 else 6
                fetchDiff = newDiff if random.random() > 0.6 else (newDiff + 2) % 6 + 1
        return questionDict[fetchDiff][random.randint(0, len(questionDict[fetchDiff]) - 1)]
    except:
        raise Exception

"""from memory_profiler import profile
@profile"""
def AnswerChecker(answer, userinput):


    try:
        ans = float(answer)
    except:
        try:
            ans = eval(answer) if isinstance(answer, str) else answer
        except Exception as e:
            mDebug.RaiseError(e)
            # Text answer
            try:
                if answer == userinput:
                    result = True
                else:
```

```
                result = False
            except:
                result = False

        else:
            # Evaluable answers (e.g. fractions)
            try:
                if type(ans) in [type(list([])), type(tuple([])), type(set([]))]:
                    check = set([eval(ansn) if isinstance(ansn, str) else ansn for ansn in ans])
                    if len(check) == 1:
                        check = list(check)[0]
                        him = eval(userinput) if isinstance(userinput, str) else userinput
                        if type(him) in [type(list([])), type(tuple([])), type(set([]))]:
                            him = list(him)[0]
                    else:
                        him = set(eval(userinput)) if isinstance(userinput, str) else set(userinput)
                    if him == check:
                        result = True
                    else:
                        result = False
                elif ans == (eval(userinput) if isinstance(userinput, str) else userinput):
                    result = True
                else:
                    result = False
            except Exception as e:
                result = False

    else:

        if int(ans) == ans:
            # Integral answers
            try:
                if int(ans) == int(eval(userinput)):
                    result = True
                else:
                    result = False
            except:
                result = False

        else:
            # Decimal answers:

            PERCENTAGEERROR = 0.5 / 100

            try:
                if abs((float(eval(userinput)) - ans) / ans) < PERCENTAGEERROR:
                    result = True
                else:
                    result = False
            except:
                result = False



    return result


def GenerateObjects(obj):

    objDict = {}

    for objName in obj.keys():

        item = obj[objName]
        objType = item["objname"].lower()
        args = item["args"].strip()

        OBJ = eval(switch[objType] + args)

        objDict[objName] = OBJ
```

```
    return objDict


def Process(objDict, string, count = 0):

    # objDict stores the objects
    # string stores the string to be processed
    # counter stores the recursion depth

    # If there are curly brackets in the string,
    # it is either the whole question or answer statmenet
    # or that it is a placeholder where there are nested placeholders that needs
    # to be evaluated
    if "{" in string or "}" in string:

        tempString = ""
        returnString = ""
        counter = 0
        inEnv = False

        # Reading the string from left to right, ...
        for character in string:

            # print(str(counter)+str(character), end=" ")

            # We enter an environment when we meet a base bracket
            # Start the environment, and write the characters to another stream, called tempString
            # If the curly bracket is not a base bracket
            # We drop the curly brackets
            # Otherwise, we include the curly brackets
            # Increasing the number of corresponding close brackets by 1
            if character == "{":

                if counter == 0:
                    pass
                else:
                    tempString += character
                counter += 1
                inEnv = True

            # When we meet a closing bracket
            # We reduce the required closing brackets by 1
            # If the closing bracket is a base bracket, and we are inside the environment
            # We leave the environment
            # We send the tempString to another Process, which will retrieve the attribute
            # and append it to the returnString
            # if there are no nesting brackets
            # The program will continue writing characters to tempString if the
            # environment is not ended
            elif character == "}":

                counter -= 1

                if counter == 0 and inEnv:

                    inEnv = False
                    # print("¥n" + tempString)

                    tempString = str(Process(objDict, tempString, count + 1))
                    # print(tempString)
                    returnString += tempString

                    tempString = ""

                if counter > 0:

                    tempString += character

            # for characters that are not curly brackets
            # we write them to tempString if we're in the enviroment
            # and to the return string if not
```

```
            else:
                if inEnv:
                    tempString += character
                else:
                    returnString += character

            # print(str(counter) + character, end=" ")

        # if the recursion depth is greater than 1,
        # then it is a placeholder with nested placeholders
        # in this case, we evaluate the placeholder
        # and include the displayOptions.
        inString = returnString
        if count > 0:
            returnString = str(AttrRetriever(objDict, returnString))
        if count == 1:
            returnString = DisplayOptions(inString, returnString)

        return returnString

    # If there are no curly brackets in the string...
    else:

        # If the recursion depth is 1 (contains display options)
        # retrieve the attributes,
        # and apply the display options
        if count == 1:
            inString = string
            returnString = str(AttrRetriever(objDict, string))
            returnString = DisplayOptions(inString, returnString)

        # If the recursion depth is greater than 1
        # just retrieves the attributes
        elif count > 1:
            returnString = str(AttrRetriever(objDict, string))

        # If for some reason, the recursion depth is 0
        # just return the string
        else:
            returnString = string

        return returnString


def DisplayOptions(inString, returnString):

    if "_hideone" in inString:
        if returnString == "1":
            string = ""
        elif returnString == "-1":
            string = "-"

    if "_showsign" in inString:
        try:
            if float(eval(returnString)) >= 0:
                string = "+" + returnString
        except:
            pass

    if "_showsignspace" in inString:
        try:
            if float(eval(returnString)) > 0:
                string = "+ " + returnString
            else:
                string = "- " + str(abs(eval(returnString)))
        except:
            pass

    try:
        return string
    except:
```

```
        return returnString


def QuestionFiller(obj, questionStm, answerStm):


    question = questionStm
    answer = answerStm
    failNo = 0


    while True:
        try:

            Shuffle()
            objCopy = obj.copy()
            objDict = GenerateObjects(objCopy)
            question = Process(objDict, question)
            answer = Process(objDict, answer)
            try:
                if "/" not in answer:
                    answer = eval(answer)
            except:
                pass

        except Exception as exception:
            del objDict
            del question
            del answer
            question = questionStm
            answer = answerStm
            failNo += 1
            if failNo >= 1000:
                raise exception
        else:
            break

    # Postprocessing
    answer = PostProcessing(answer)

    return question, answer


def AttrRetriever(objDict, argsString):

    specList = re.split(",", argsString)

    obj = objDict[specList[0]]
    argsList = specList[1:]

    sysArgs = 0
    tempArgs = []
    for args in argsList[1:]:
        if "_" not in args:  # if _ is in args, then it specifies a display option
            tempArgs.append(args)
        else:
            sysArgs += 1

    if (len(argsList[1:]) - sysArgs == 0 and sysArgs > 0) or len(argsList) == 1:
        coeff = getattr(obj, argsList[0])
    else:
        coeff = ExecuteFunction(obj, argsList[0], tempArgs)

    return str(coeff)


def PostProcessing(answer):

    try:
        if type(answer) == type(""):
            if "/" in answer:
```

```
            a, b = re.split("/", answer)
            if b == "1":
                newAnswer = a
            else:
                hcf = HCF(a,b)
                if hcf != 1:
                    newAnswer = str(a//hcf)+"/"+str(b//hcf)
                else:
                    newAnswer = answer
        else:
            newAnswer = answer
    else:
        newAnswer = answer
except:
    newAnswer = answer

try:
    if isinstance(answer, float):
        newAnswer = f'{answer:.4g}'
    else:
        newAnswer = answer
except:
    newAnswer = answer

if isinstance(newAnswer, str):
    if "¥uff0c" in newAnswer:
        newAnswer.replace("¥uff0c", ",")

try:
    lst = eval(newAnswer)
    if type(lst) in [type(list([])), type(tuple([])), type(set([]))]:
        newAnswer = list(set(lst))
except:
    pass

try:
    if isinstance(answer, Fraction):
        newAnswer = str(answer.numerator) + "/" + str(answer.denominator)
except:
    pass

try:
    return newAnswer
except:
    return answer


def main():

    def DisplayDictList(d):
        for i in d.keys():
            print(str(i) + ' : ', end="")
            print(d[i])

    questionDict = ReadQuestion("files/Questions.json")

    while True:
        question = QuestionGenerator(random.randint(1, 6), questionDict)
        DisplayDictList(question)

    for k in range(6):
        j = 0
        run = True
        while run:
            try:

                # For each question, repeat five times
                for i in range(5):

                    # Generates a question
                    question = QuestionGenerator(k+1, questionDict, j)
```

```
                # Print out the question
                print(question["comment"] + ": " + question["question"])

                # Prompt for input
                ans = input()

                # Checks the answer and display the results
                print(AnswerChecker(question["answer"], ans), question["answer"])

                print()
            j += 1
        except:
            run = False


if __name__ == "__main__":
    main()
```

31. *mQuestionsWorkplace.py*

```
"""
mQuestionWorkplace.py
"""


from mQuestions import *


def main():

    def DisplayDictList(d):
        for i in d.keys():
            print(str(i) + ' : ', end="")
            print(d[i])

    questionDict = ReadQuestion("files/TestQuestions.json")

    answer = "3/2"
    ans = 3/2
    print(AnswerChecker(answer, ans), answer)

    q = QuestionFetch(6, questionDict, 0)
    print()
    print(QuestionFiller(q["obj"], q["questionStm"], q["answerStm"]))

    """for k in range(6):
        while True:
            try:
                question = QuestionGenerator(k+1, questionDict, 0)
                print(question["comment"] + ": " + question["question"])
                ans = input()
                print(AnswerChecker(question["answer"], ans), question["answer"])
                print()
            except Exception as e:
                print (e)
                break"""

    for k in range(6):
        j = 0
        run = True
        while run:
            try:
                for i in range(5):
                    question = QuestionGenerator(6, questionDict, j)
                    print(question["comment"] + ": " + question["question"])
                    ans = input()
                    print(AnswerChecker(question["answer"], ans), question["answer"])
                    print()
                j += 1
            except Exception as e:
                run = False

main()
```

32. pyAnimation.py

```
"""
pyAnimation.py
"""

import pygame
FadeInConst = 1.08
FadeOutConst = 1.08

def FadeOut(screen, clock, fps):

    surf = pygame.Surface((screen.get_size()))
    alpha = 2

    while alpha < 255:
        pygame.event.get()
        alpha = min(alpha**FadeOutConst, 255)
        surf.set_alpha(int(alpha))
        screen.blit(surf, (0, 0))
        pygame.display.update()
        clock.tick(fps)

def FadeIn(screen, displayFunction, clock, fps):

    surf = pygame.Surface((screen.get_size()))
    alpha = 255

    while alpha > 0:
        pygame.event.get()
        alpha = max(0, alpha//FadeInConst)
        surf.set_alpha(int(alpha))
        # Blits the screenshot onto the screen
        displayFunction()
        # Blits the half-transparent black overlay
        screen.blit(surf, (0, 0))
        pygame.display.update()
        clock.tick(fps)

def EnterFade(screen, displayFunction, clock, fps, waitTime = 50):

    FadeOut(screen, clock, fps)
    pygame.time.delay(waitTime)
    FadeIn(screen, displayFunction, clock, fps)

if __name__ == "__main__":

    screen = pygame.display.set_mode((500, 500))
    screen.fill((255, 255, 255))
    surf = pygame.Surface((300, 300))
    surf.set_alpha(0)
    screen.blit(surf, (0, 0))
    pygame.display.update()
    while True:
        screen.fill((255, 255, 255))
        FadeOut(screen)

    input()
```

33. *pyFloating.py*

```
"""
pyFloating.py
"""

import pygame
import pyImage
import math


class Floating:

    def __init__(self, floater, dim, clock, period = 1.5): # period in seconds

        self.height = dim[1]
        self.width = dim[0]
        self.floater = pyImage.ScaleGraphics(floater, self.width, True, False)
        self.floaterHeight = self.floater.get_height()
        self.floatHeight = self.height - self.floaterHeight
        self.y = 0
        self.angularFrequency = 2 * math.pi / period / 1000 # in rad ms^-1
        self.time = 0
        self.clock = clock
        # y = cos(wt), where w is in ms^-1 and t is in ms

    def Reset(self):
        self.time = 0

    def Update(self):
        self.time += self.clock.get_time()
        self.y = self.floatHeight * (math.sin(self.angularFrequency * self.time) + 1)/2
        tempSurf = pygame.Surface((self.width, self.height), pygame.SRCALPHA)
        tempSurf.blit(self.floater, (0, self.y))
        return tempSurf
```

34. pyImage.py

```
"""
pyImage.py
"""


import pygame

def ScaleGraphics(graphics, metric, xbool, ybool):

    if xbool and not ybool:
        ratioCoeff = metric / graphics.get_width()
        return pygame.transform.scale(graphics, (metric, int(ratioCoeff * graphics.get_height())))
    elif not xbool and ybool:
        ratioCoeff = metric / graphics.get_height()
        return pygame.transform.scale(graphics, (int(ratioCoeff * graphics.get_width()), metric))
    else:
        print("Error occurred in ScaleGraphics")

# Not used
def SmoothScaleGraphics(graphics, metric, xbool, ybool):

    if xbool and not ybool:
        ratioCoeff = metric / graphics.get_width()
        return pygame.transform.smoothscale(graphics, (metric, int(ratioCoeff * graphics.get_height
()))
    elif not xbool and ybool:
        ratioCoeff = metric / graphics.get_height()
        return pygame.transform.smoothscale(graphics, (int(ratioCoeff * graphics.get_width()), metr
ic))
    else:
        print("Error occurred in ScaleGraphics")

def Crop(image, position, dimensions):

    Surf = pygame.Surface(dimensions, pygame.SRCALPHA)
    Surf.blit(image, (-position[0], -position[1]))
    return Surf

def Wallpaper(image, size):

    a = ScaleGraphics(image, size[1], False, True)
    if a.get_width() < size[0]:
        k = pygame.Surface(size)
        k.fill((255, 255, 255))
        k.blit(a, (int((size[0] - a.get_width())/2), 0))
        a = k
    else:
        a = Crop(a, (int((a.get_width() - size[0])/2), 0), size)
    return a
```

35. pyInput.py

```
"""
pyInput.py
"""

import pyText
import pygame
from pygame.locals import *
import GlobalVar

# Comment: one of the biggest disadvantage of pygame (and python) is that
# in game development, engines have to be developed yourself, and it is
# in fact very fustrating and time consuming. Therefore, professional game
# developers choose to use other languages like Java and C, or web
# languages like JavaScript to write games.


'''
class TextInput

TextInput(font, width, clock,
              color = (0,0,0),
              mode = "text",
              initialText = '',
              cursorBlinkTime = 500,
              typeFirstTime = 500,
              typeTime = 50,
              cursorThickness = 1,
              cursorColor = (0, 0, 0)):
Creates a TextInput object.

Surface()
Returns the surface of the text input object.

Debug()
Prints the cursor position and the text stored.

Typing()
If placed in a loop, it starts accepting edits from the player.

Text()
Returns the text stored in the text input object.


'''

class TextInput:

    def __init__ (self, font, width, clock,
                  color = (0,0,0),
                  initialText = '',
                  cursorBlinkTime = 500,
                  typeFirstTime = 1000,
                  typeTime = 70,
                  cursorThickness = 1,
                  cursorColor = (0, 0, 0)):

        # Argument assignment
        self.height = font.get_height()
        self.width = width
        self.font = font
        self.textSurface = pygame.Surface((self.width, self.font.get_height()), pygame.SRCALPHA)
        self.color = color

        # Determines if the player can write text anymore
        self.textFlag = True

        # My own clock
        self.clock = clock
```

```
        # Text Related
        self.text = initialText
        self.cursorPos = 0   # len(self.text) is maximum

        # Cursor Related
        self.cursorBlinkTime = cursorBlinkTime
        self.cursorClock = 0
        self.showCursor = True

        self.cursorThickness = cursorThickness
        self.cursor = pygame.Surface((self.cursorThickness, self.height))
        self.cursorColor = cursorColor
        self.cursor.fill(cursorColor)

        # Key Repetition
        self.typeFirstTime = typeFirstTime
        self.typeTime = typeTime
        self.typeClock = 0
        self.typeFirst = False
        self.typeHeld = None

    def Surface(self):
        return self.textSurface

    def Text(self):
        return self.text

    def addText(self, char):
        self.text = self.text[:self.cursorPos] + char + self.text[self.cursorPos:]
        self.cursorPos += 1

    def backSpace(self):
        self.text = self.text[:self.cursorPos - 1] + self.text[self.cursorPos:]
        self.cursorPos = max(0, min(self.cursorPos - 1, len(self.text)))

    def delete(self):
        self.text = self.text[:self.cursorPos] + self.text[self.cursorPos + 1:]

    def left(self):
        self.cursorPos = max(0, self.cursorPos - 1)

    def right(self):
        self.cursorPos = min(self.cursorPos + 1, len(self.text))

    def holdTest(self):
        if self.typeHeld == 'backspace':
            self.backSpace()
        elif self.typeHeld == 'delete':
            self.delete()
        elif self.typeHeld == 'left':
            self.left()
        elif self.typeHeld == 'right':
            self.right()
        elif self.textFlag:
            self.addText(self.typeHeld)

    def Debug(self):
        print("cursor:", self.cursorPos)
        print("text:", self.text)

    def EventCheck(self, event):

        if event.type == KEYDOWN:
            if event.key == K_RETURN:
                return True

            elif event.key == K_DELETE:
                self.delete()
                self.typeHeld = "delete"

            elif event.key == K_BACKSPACE:
```

```
                self.backSpace()
                self.typeHeld = "backspace"

            elif event.key == K_LEFT:
                self.left()
                self.typeHeld = "left"

            elif event.key == K_RIGHT:
                self.right()
                self.typeHeld = "right"

            else:
                if event.key not in (K_BACKSPACE, K_RETURN, K_TAB, K_CLEAR, ¥
                                     K_PAUSE, K_ESCAPE, K_DELETE, K_UP, K_DOWN, K_RIGHT, ¥
                                     K_LEFT, K_INSERT, K_HOME, K_END, ¥
                                     K_PAGEUP, K_PAGEDOWN, K_RSHIFT, K_LSHIFT, ¥
                                     K_RCTRL, K_LCTRL, K_RALT, K_LALT):
                    char = event.unicode
                    if self.textFlag:
                        # Adds text
                        self.addText(char)
                        self.typeHeld = char

        elif event.type == KEYUP:
            self.typeFirst = False
            self.typeClock = 0
            self.typeHeld = None

    def UpdateCursor(self):

        self.cursorClock += self.clock.get_time()
        if self.cursorClock > self.cursorBlinkTime:
            self.cursorClock = self.cursorClock % self.cursorBlinkTime
            self.showCursor = not self.showCursor
        self.UpdateScreen()
        self.TypeHeld()

    def UpdateScreen(self):

        self.textSurface = pygame.Surface((self.width, self.font.get_height()), pygame.SRCALPHA)

        posCounter = self.cursorThickness

        if len(self.text) == 0:
            if self.showCursor:
                rect2 = self.cursor.get_rect()
                rect2.centery = int(self.textSurface.get_height() / 2)
                rect2.left = posCounter
                self.textSurface.blit(self.cursor, rect2)
            self.font.set_italic(True)
            self.textSurface.blit(self.font.render("Type Here", 1, pygame.Color("gray")), (posCount
er, 0))
            self.font.set_italic(False)

        for counter in range(len(self.text)):
            char = self.text[counter]
            charWidth = self.font.size(char)[0]
            if posCounter + charWidth + self.cursorThickness < self.width:
                self.textSurface.blit(self.font.render(char, 1, self.color), (posCounter, 0))
                posCounter += charWidth
                if counter == self.cursorPos - 1 and self.showCursor:
                    rect2 = self.cursor.get_rect()
                    rect2.centery = int(self.textSurface.get_height() / 2)
                    rect2.left = posCounter
                    self.textSurface.blit(self.cursor, rect2)
                posCounter += self.cursorThickness
            else:
                self.text = self.text[:counter]
                self.cursorPos = min(self.cursorPos, len(self.text))
                break
        if posCounter + self.font.size("M")[0] > self.width:
```

```
            self.textFlag = False
        else:
            self.textFlag = True

    def TypeHeld(self):

        try:
            if self.typeHeld != None:
                keys = pygame.key.get_pressed()
                if True not in keys:
                    self.typeHeld = None
        except Exception as e:
            pass

        if self.typeHeld != None:
            self.typeClock += self.clock.get_time()
            if self.typeClock > self.typeFirstTime and self.typeFirst == False:
                self.typeClock = 0
                self.typeFirst = True
                self.holdTest()
            elif self.typeClock > self.typeTime and self.typeFirst:
                self.typeClock %= self.typeTime
                self.holdTest()

    def Typing(self, event = None):

            if event:
                self.EventCheck(event)
                self.UpdateScreen()
            else:
                for event in pygame.event.get():
                    GlobalVar.Quit(event)
                    if self.EventCheck(event):
                        self.UpdateScreen()
                        return True
                self.UpdateScreen()
                self.UpdateCursor()
                self.TypeHeld()




if __name__ == "__main__":
    pass

    pygame.init()
    pygame.font.init()

    # Screen
    ScreenWidth = int(1280 / 1.5)
    ScreenHeight = int(960 / 1.5)
    Screen = pygame.display.set_mode((ScreenWidth, ScreenHeight))

    # Test Font
    BasicFont = pygame.font.Font(GlobalVar.GameFont, 20)

    clock = pygame.time.Clock()
    s = TextInput(BasicFont, 500, clock,  color = (133, 0, 0), cursorColor = (200, 100, 200))

    while True:

        if s.Typing():
            my = s.Text()
            break

        # s.Debug()
        Screen.fill((0,0,0))
        Screen.blit(s.textSurface,(0, 100))
```

```
    keys = pygame.key.get_pressed()

    pygame.display.flip()
    clock.tick(60)

pygame.quit()

print(my)
```

36. *pyMenu.py*

```
"""
pyMenu.py
"""


import pygame
from pygame.locals import *
import pyImage
import pyText
import GlobalVar
import pyInput
import pyMouse


class SelectionMenu:

    def __init__(self, scroller, background, scrollerShift = 3, textShift = 3, alignment = "left",
spacing = 0, scrollerSize = -1):

        # Initiate variables
        self.itemList = []
        self.scroller = scroller
        self.scrollerShift = scrollerShift
        self.background = GlobalVar.DefaultMenuBackground
        self.selectedBackground = GlobalVar.DefaultMenuSelectBackground

        self.width = background[0] # x
        self.height = background[1] # y
        self.size = background # size
        self.textShift = textShift
        self.spacing = spacing
        self.alignment = alignment

        if scrollerSize != -1:
            self.scrollerSize = scrollerSize
        else:
            self.scrollerSize = self.height

        # Edit scroller and create scrollerSurf
        self.scroller = pygame.transform.scale(self.scroller, (self.scrollerSize - self.scrollerShi
ft * 2, self.scrollerSize - self.scrollerShift * 2))
        self.scrollerSurf = pygame.Surface((self.scrollerSize, self.scrollerSize), pygame.SRCALPH
A)
        self.scrollerRect = self.scroller.get_rect()
        self.scrollerRect.center = (int(self.scrollerSize / 2), int(self.scrollerSize / 2))
        self.scrollerSurf.blit(self.scroller, self.scrollerRect)

        # Initiate counter
        self.counter = 0
        self.selected = -1

    def CreateSurf(self, text, font, position, name = '', color = (0, 0, 0), multiline = False, scro
llerPos = "left"):

        # Assign textSurf to text image if image is used instead of text.
        if multiline:
            textList = pyText.TextWrap(text, font, self.width, 0, color)
            textSurf = pyText.TextRender(textList, font, self.spacing, self.alignment, color)
        else:
            textSurf = font.render(text, 1, color)

        tempRect = textSurf.get_rect()
        tempRect.centery = int(self.height / 2)
        if self.alignment == "left":
            tempRect.left = int(self.textShift)
        elif self.alignment == "center":
            tempRect.centerx = int(self.width / 2)
        elif self.alignment == "right":
            tempRect.right = int(self.width - self.textShift)
```

```
        else:
            print("Error occurred in CreateItem")

        size = textSurf.get_size()
        tempBg = pygame.Surface(self.size, pygame.SRCALPHA)

        selectedBg = tempBg.copy()
        selectedBg.blit(pygame.transform.scale(self.selectedBackground, size), tempRect)
        tempBg.blit(pygame.transform.scale(self.background, size), tempRect)

        tempBg.blit(textSurf, tempRect)
        selectedBg.blit(textSurf, tempRect)

        tempDict = {}
        if name == '':
            tempDict['name'] = text
        else:
            tempDict['name'] = name
        tempDict['pos'] = position
        tempDict['surface'] = tempBg
        tempDict['selected'] = selectedBg
        tempDict['scrollerPos'] = scrollerPos


        return tempDict

    def CreateItem(self, text, font, position, name='', color=(0, 0, 0), multiline=False, scrollerP
os = "left"):

        tempDict = self.CreateSurf(text, font, position, name, color, multiline,scrollerPos)

        self.itemList.append(tempDict)

        return len(self.itemList) - 1

    def ChangeItem(self, index, text, font, position, name='', color=(0, 0, 0), multiline=False, sc
rollerPos = "left"):

        tempDict = self.CreateSurf(text, font, position, name, color, multiline, scrollerPos)

        try:
            self.itemList[index] = tempDict
        except:
            print("Error occured in SelectionMenu.ChangeItem")

        return len(self.itemList) - 1

    def RemoveItem(self, index):

        try:
            del(self.tempDict[index])
        except:
            print("Error occured in SelectionMenu.RemoveItem")

        return len(self.itemList) - 1

    def ChangePos(self, pos, index = -1, name = ""):

        if index != -1 and not name:
            self.itemList[index]['pos'] = pos
        elif index == -1 and name:
            for i in self.itemList:
                if i["name"] == name:
                    i["pos"] = pos
                    break
        elif index != -1 and self.itemList[index]["name"] == name:
            self.itemList[index]["name"] = pos
        else:
            print("Errored occurred in ChangePos.")

    def ScrollItem(self, direction=1):
```

```
        self.counter = (self.counter + direction + len(self.itemList)) % len(self.itemList)
        return self.counter

    def Clicking(self, event, offset = (0, 0)):

        if event.type == MOUSEMOTION:
            for num, i in enumerate(self.itemList):
                if pyMouse.InArea(event, i["pos"], self.size, offset):
                    self.selected = num
                    self.counter = num
                    break
                else:
                    self.selected = -1
        elif event.type == MOUSEBUTTONDOWN:
            if event.button == 1:
                if self.selected == -1:
                    return None
                else:
                    try:
                        GlobalVar.ButtonClick.play()
                    except:
                        pass
                    return self.selected

    def Scrolling(self, event, offset = (0, 0)):

        if event.type == KEYDOWN:
            if event.key == K_DOWN:
                self.ScrollItem(1)
            elif event.key == K_UP:
                self.ScrollItem(-1)
            elif event.key == K_RETURN:
                try:
                    GlobalVar.ButtonClick.play()
                except:
                    pass
                return self.counter
        return self.Clicking(event, offset)

    def GetSize(self, index=-1, name=""):

        if index != -1 and not name:
            return self.itemList[index]["surface"].get_size()
        elif index == -1 and name:
            for i in self.itemList:
                if i["name"] == name:
                    return i["surface"].get_size()
        elif index != -1 and self.itemList[index]["name"] == name:
            return self.itemList[index]["surface"].get_size()
        else:
            print("Errored occurred in ChangePos.")

    def GetSelected(self):
        return self.selected

    def DisplayItem(self, screen):

        for i in range(len(self.itemList)):
            scrollerPos = self.itemList[i]['scrollerPos']
            if scrollerPos == "below":
                if i == self.counter:
                    screen.blit(self.scrollerSurf, (self.itemList[i]['pos'][0] + int((self.GetSize(i
ndex = i)[0] - self.scrollerSize)/2), self.itemList[i]['pos'][1] + self.height))
                pos = self.itemList[i]['pos']
            elif scrollerPos == "above":
                if i == self.counter:
                    screen.blit(self.scrollerSurf, (self.itemList[i]['pos'][0] + int((self.GetSize(i
ndex = i)[0] - self.scrollerSize)/2), self.itemList[i]['pos'][1]))
                pos = (self.itemList[i]['pos'][0], self.itemList[i]['pos'][1] + self.scrollerSize)
            elif scrollerPos == "left":
```

```
            if i == self.counter:
                tempRect = self.scrollerSurf.get_rect()
                tempRect.centery = int(self.height/2)+self.itemList[i]['pos'][1]
                tempRect.left = self.itemList[i]['pos'][0]
                screen.blit(self.scrollerSurf, tempRect)
            pos = (self.itemList[i]['pos'][0] + self.scrollerSize, self.itemList[i]['pos'][1])
        elif scrollerPos == "right":
            if i == self.counter:
                tempRect = self.scrollerSurf.get_rect()
                tempRect.centery = int(self.height/2)+self.itemList[i]['pos'][1]
                tempRect.left = self.itemList[i]['pos'][0] + self.GetSize(index = i)[0]
                screen.blit(self.scrollerSurf, tempRect)
            pos = self.itemList[i]['pos']
        else:
            print("Error occurred in DisplayItem: invalid argument scrollerPos = ", scrollerPos)
        screen.blit(self.itemList[i]['surface'] if self.selected != i else self.itemList[i]['selected'], pos)


    def Debug(self):
        print("No. of Items: ", len(self.itemList))
        for i in range(len(self.itemList)):
            print(f'{i:<5}{self.itemList[i]["name"]:<20}{self.itemList[i]["pos"]}')

    def GetCounter(self):
        return self.counter

    def SetCounter(self, counter):
        self.counter = counter
        return self.counter

class FrameMenu:

    # Originally originated from the making of a_Stats in AMenu

    HeightTotal = 21.0
    WidthTotal = 29.7

    # Text Related
    TopCoord = 2.1 / HeightTotal
    TitleFont = 1.4 / HeightTotal
    HeaderFont = 0.9 / HeightTotal
    SubHeaderFont = 0.75 / HeightTotal
    TextFont = 0.65 / HeightTotal
    ExplanationFont = 0.50 / HeightTotal
    TitleSpacing = 1.6 / HeightTotal
    HeaderSpacing = 0.6 / HeightTotal
    SubHeaderSpacing = 0.4 / HeightTotal
    TextSpacing = 0.30 / HeightTotal
    ExplanationSpacing = 0.20/HeightTotal
    TextHorzShift = 3.2 / WidthTotal

    # Image Related
    GraphicsSize = 9.7 / HeightTotal
    GraphicsVertShift = 5.0 / HeightTotal
    GraphicsHorzShift = 15.4 / WidthTotal

    ScrollerSize = 1.4 / HeightTotal
    MenuTextSize = 1.1 / HeightTotal
    MenuHeight = 1.4 / HeightTotal
    MenuSpacing = 0.3 / HeightTotal
    MenuWidth = 20 / WidthTotal
    MenuTextShift = 0.3 / WidthTotal

    def __init__(self, screen, clock = None):

        self.screen = screen
        self.height = self.screen.get_height()
        self.width = self.screen.get_width()

        self.topCoord = int(self.TopCoord * self.height)
```

```
        self.titleFontSize = int(self.TitleFont * self.height)
        self.headerFontSize = int(self.HeaderFont * self.height)
        self.textFontSize = int(self.TextFont * self.height)
        self.explanationFontSize = int(self.ExplanationFont * self.height)
        self.menuTextSize = int(self.MenuTextSize * self.height)
        self.subHeaderFontSize = int(self.HeaderFont * self.height)

        self.titleFont = pygame.font.Font(GlobalVar.DefaultMenuTitleFontString, self.titleFontSiz
e)
        self.headerFont = pygame.font.Font(GlobalVar.DefaultMenuHeaderFontString, self.headerFontS
ize)
        self.textFont = pygame.font.Font(GlobalVar.DefaultMenuTextFontString, self.textFontSize)
        self.explanationFont = pygame.font.Font(GlobalVar.DefaultMenuExplanationFontString, self.e
xplanationFontSize)
        self.menuFont = pygame.font.Font(GlobalVar.DefaultMenuMenuFontString, self.menuTextSize)
        self.subHeaderFont = pygame.font.Font(GlobalVar.DefaultMenuHeaderFontString, self.subHeade
rFontSize)
        self.inputFontString = GlobalVar.DefaultInputFontString

        self.titleSpacing = int(self.TitleSpacing * self.height)
        self.headerSpacing = int(self.HeaderSpacing * self.height)
        self.textSpacing = int(self.TextSpacing * self.height)
        self.subHeaderSpacing = int(self.SubHeaderSpacing * self.height)
        self.explanationSpacing = int(self.ExplanationSpacing * self.height)
        self.textHorzShift = int(self.TextHorzShift * self.width)

        self.menuSpacing = int(self.MenuSpacing * self.width)
        self.scrollerSize = int(self.ScrollerSize * self.height)

        self.menuHeight = int(self.MenuHeight * self.height)
        self.menuWidth = int(self.MenuWidth * self.width)
        self.menuTextShift = int(self.MenuTextShift * self.width)

        self.graphics = None
        self.graphicsSize = int(self.GraphicsSize * self.height)
        self.graphicsHorzShift = int(self.GraphicsHorzShift * self.width)
        self.graphicsVertShift = int(self.GraphicsVertShift * self.height)
        self.graphicsRect = (self.graphicsHorzShift, self.graphicsVertShift)

        self.inputSpacing = self.textSpacing
        self.textWidth = self.graphicsHorzShift - self.textHorzShift

        self.menu = None
        self.input = []
        self.ret = False

        if clock:
            self.clock = clock
        else:
            self.clock = pygame.time.Clock()

        self.blitList = []

        self.tempTop = self.topCoord

    def TextWrap(self, text, Header = False, Title = False, Text = False, Explanation = False, Subhe
ader = False, color = (0, 0, 0), graphics = False):

        if graphics:
            self.textWidth = self.graphicsHorzShift - self.textHorzShift
        else:
            if self.graphics:
                self.textWidth = self.graphicsHorzShift - self.textHorzShift
            else:
                self.textWidth = self.width - self.textHorzShift*2

        if Title:
            font = self.titleFont
        elif Header:
            font = self.headerFont
        elif Text:
```

```
            font = self.textFont
        elif Explanation:
            font = self.explanationFont
        elif Subheader:
            font = self.subHeaderFont
        else:
            print("An error has occurred in TextWrap")
            return

        textList = pyText.TextWrap(text, font, self.textWidth)

        for i in textList:
            self.InsertText(i, Header = Header, Text = Text, Explanation = Explanation, Subheader =
Subheader, Title = Title, color = color)

    def AddGraphics(self, graphics):

        if graphics:
            if isinstance(graphics, str):
                self.graphics = pygame.image.load(graphics)
            else:
                self.graphics = graphics
            self.graphics = pyImage.ScaleGraphics(self.graphics, self.graphicsSize, False, True)
        else:
            self.graphics = None

    def InitiateMenu(self):

        if self.menu:
            pass
        else:
            myScroller = GlobalVar.DefaultMenuScroller
            myBackground = (self.menuWidth, self.menuHeight)

            self.menu = SelectionMenu(myScroller, myBackground, textShift = self.menuTextShift, scro
llerSize = self.scrollerSize)

    def InsertText(self, text, color = (0, 0, 0), Header = False, Text = False, Explanation = False,
 Title = False, Subheader = False, Enter = False):

        if Title:
            if text:
                tempSurf = self.titleFont.render(text, 1, color)
                self.blitList.append([tempSurf, (self.textHorzShift, self.tempTop)])
            self.tempTop += self.titleFontSize + self.titleSpacing
        elif Header:
            if text:
                tempSurf = self.headerFont.render(text, 1, color)
                self.blitList.append([tempSurf, (self.textHorzShift, self.tempTop)])
            self.tempTop += self.headerFontSize + self.headerSpacing
        elif Text:
            if text:
                tempSurf = self.textFont.render(text, 1, color)
                self.blitList.append([tempSurf, (self.textHorzShift, self.tempTop)])
            self.tempTop += self.textFontSize + self.textSpacing
        elif Explanation:
            if text:
                tempSurf = self.explanationFont.render(text, 1, color)
                self.blitList.append([tempSurf, (self.textHorzShift, self.tempTop)])
            self.tempTop += self.explanationFontSize + self.explanationSpacing
        elif Subheader:
            if text:
                tempSurf = self.subHeaderFont.render(text, 1, color)
                self.blitList.append([tempSurf, (self.textHorzShift, self.tempTop)])
            self.tempTop += self.subHeaderFontSize + self.subHeaderSpacing
        elif Enter:
            self.tempTop += self.titleFontSize + self.textSpacing*2
        else:
            print("An error has occurred in InsertText")

    def InsertMenu(self, text, pos = None):
```

```
        if self.menu != None:
            if pos:
                return self.menu.CreateItem(text, self.menuFont, pos)
            else:
                index = self.menu.CreateItem(text, self.menuFont, (self.textHorzShift, self.tempTo
p))
                self.tempTop += self.menuSpacing + self.menuHeight
                return index
        else:
            print("Menu is not initialized!")

    def Blit(self):

        for surf in self.blitList:
            self.screen.blit(surf[0], surf[1])
        if self.menu:
            self.menu.DisplayItem(self.screen)
        if self.graphics:
            self.screen.blit(self.graphics, (self.graphicsHorzShift, self.graphicsVertShift))
        for input in self.input:
            self.screen.blit(input["textPrompt"], input["textRect"])
            self.screen.blit(input["inputObj"].Surface(), input["inputRect"])

    def Scrolling(self, event):
        if self.input:
            rep = self.menu.Scrolling(event)
            if self.menu.GetSelected() not in [-1, 0]:
                self.ret = True
            elif self.ret and self.menu.GetSelected() == -1:
                self.menu.SetCounter(0)
                self.ret = False
            return rep
        else:
            return self.menu.Scrolling(event)

    def UpdateCursor(self):

        for input in self.input:
            if self.menu.GetCounter() == input["index"]:
                return input["inputObj"].UpdateCursor()

    def Typing(self, event = None):
        for input in self.input:
            if self.menu.GetCounter() == input["index"]:
                return input["inputObj"].Typing(event)

    def Text(self):
        for input in self.input:
            if self.menu.GetCounter() == input["index"]:
                return input["inputObj"].Text()

    def Counter(self):
        return self.menu.GetCounter()

    def InsertInput(self, promptText, promptColor=pygame.Color("Black"),
                    inputColor=pygame.Color("Black"), Header = False, Title = False, Text = False):

        if self.menu != None:
            pass
        else:
            self.InitiateMenu()

        if Header:
            font = self.headerFont
            size = self.headerFontSize
            spacing = self.headerSpacing
        elif Title:
            font = self.titleFont
            size = self.titleFontSize
            spacing = self.titleSpacing
```

```
        elif Text:
            font = self.textFont
            size = self.textFontSize
            spacing = self.textSpacing
        else:
            print("Error occured in InsertInput!")
            return

        textPrompt = font.render(promptText, 1, promptColor)
        textRect = (self.textHorzShift, self.tempTop)
        textWidth = textPrompt.get_width()

        inputFont = pygame.font.Font(self.inputFontString, size)
        inputObj = pyInput.TextInput(inputFont,
                                self.width - self.textHorzShift - 2 * self.inputSpacing - textWidth,
                                self.clock, inputColor)
        inputRect = (self.textHorzShift + textWidth + self.inputSpacing,
                    self.tempTop)

        index = self.InsertMenu("", (self.width, self.height))

        self.tempTop += size + spacing

        self.input.append(
            {
                "textPrompt": textPrompt,
                "textRect": textRect,
                "inputObj": inputObj,
                "inputRect": inputRect,
                "index": index
            }
        )

        return index

    def DisplayFunctionSurf(self):

        tempSurf = pygame.Surface(self.screen.get_size())
        tempSurf.fill((255, 255, 255))

        tempCopy = FrameMenu(tempSurf)

        tempCopy.graphics = self.graphics
        tempCopy.menu = self.menu
        tempCopy.input = self.input
        tempCopy.blitList = self.blitList

        if self.menu:
            tempCopy.menu.SetCounter(self.menu.GetCounter())

        tempCopy.Blit()

        return tempSurf



if __name__ == "__main__":

    def displayList(list):
        for i in list:
            print(i)

    pygame.init()
    pygame.font.init()

    # Screen
    ScreenWidth = int(1280/1.5)
    ScreenHeight = int(960/1.5)
    Screen = pygame.display.set_mode((ScreenWidth, ScreenHeight))
    TextShift = 25
    TextHeight = 200
```

```
    Screen.fill((241, 43, 32))
    # Test Font
    BasicFont = pygame.font.Font(GlobalVar.DefaultMenuMenuFontString, 20)

    something = pyText.TextWrap("""¥u12516¥u12531¥u12487¥u12524Simply put, I hate you.""", BasicFo
nt, ScreenWidth - 2 * TextShift, TextShift)
    displayList(something)
    abcde = pyText.TextRender(something, BasicFont, 10, color = (255,0,0))

    BasicBackground = (100, 40)
    Scroller = pygame.Surface((40, 40), pygame.SRCALPHA)
    pygame.draw.circle(Scroller, (0, 255, 0), (20, 20), 20)
    menu = SelectionMenu(Scroller, BasicBackground, 6)
    menu.CreateItem("Choice 1", BasicFont, (20, 20), color=(0, 250, 0))
    menu.CreateItem("Choice 2", BasicFont, (20, 70), color=(0, 250, 0))
    menu.CreateItem("Choice 3", BasicFont, (20, 120), color=(0, 250, 0))
    menu.Debug()

    menu.ChangePos((50, 500), name = "Choice 2")

    while True:

        Screen.fill((241, 43, 32))

        for event in pygame.event.get():
            GlobalVar.Quit(event)
            print(menu.Scrolling(event))


        Screen.blit(abcde, (TextShift * 2, ScreenHeight - TextHeight))
        menu.DisplayItem(Screen)
        pygame.display.flip()

    pygame.quit()
```

37. pyMouse.py

```
"""
pyMouse.py
"""

from pygame.locals import *


def InArea(event, start, size, offset):

    if event.type == MOUSEMOTION:
        pos = event.pos
        if start[0] + offset[0] <= pos[0] <= start[0] + size[0] + offset[0] ¥
               and start[1] + offset[1] <= pos[1] <= start[1] + size[1] + offset[1]:
            return True
    return False
```

38. pyText.py

```
"""
pyText.py
"""

"""
TextWrap(text, font, width, shift, color = (0, 0, 0)) --> List
TextRender(lines, font, spacing, alignment = 'left', color = (0, 0, 0)) --> Surface
"""

'''
Pending:
- Add text formatting
'''

import pygame

# TextWrap(text, font, width, shift = 0, color = (0, 0, 0))
def TextWrap(text, font, width, shift = 0, color = (0, 0, 0)):

    counter = 0
    displayWidth = width - 2 * shift
    words = text.split(" ")
    widthCounter = 0
    lines = [[]]
    newLine = False
    spaceWidth = font.size(' ')[0]

    for i in words:
        newLine = '¥n' in i
        # For newline characters
        if newLine:
            temp2 = i.split("¥n")
            j = temp2[0]
        else:
            j = i

        nowWidth = font.size(j)[0]
        if widthCounter + nowWidth + spaceWidth >= displayWidth:
            # Newline
            lines.append([])
            # Reset widthCounter
            widthCounter = nowWidth + spaceWidth
        else:
            # Add to width
            widthCounter += nowWidth + spaceWidth
        # Add word
        lines[-1].append(j + ' ')

        if newLine:
            if len(temp2) == 2:
                lines.append([temp2[-1] + ' '])
                widthCounter = font.size(temp2[-1] + ' ')[0]
            else:
                for k in temp2[1:]:
                    lines.append([k + ' '])
                    widthCounter = font.size(k + ' ')[0]

    for i in range(len(lines)):
        lines[i] = ''.join(lines[i])
        lines[i] = lines[i].rstrip(' ')

    return lines

# TextRender(lines, font, spacing, alignment = 'left', color = (0, 0, 0))
def TextRender(lines, font, spacing, alignment = 'left', color = (0, 0, 0)):

    # Initiation
    renderline = []
    renderwidth = 0
```

```
    renderheight = 0

    # Reading Lines
    # Generating Surfaces
    # Calculating Surface Dimensions
    for i in lines:
        k = font.render(i, 1, color)
        renderline.append(k)
        w = k.get_width()
        if w > renderwidth:
            renderwidth = w
        renderheight += k.get_height()
    renderheight += spacing * (len(renderline) - 1)

    # Create Surface, set Per-pixel Alpha
    textSurf = pygame.Surface((renderwidth, renderheight), pygame.SRCALPHA, 32)
    # Surface alpha: textSurf.set_alpha(256)
    # Colorkey: textSurf.set_colorkey((0, 0, 0))

    # Blit text on Surface
    topCoord = 0
    for i in renderline:
        tempRect = i.get_rect()
        tempRect.top = topCoord
        if alignment == 'left':
            tempRect.left = 0
        elif alignment == 'center':
            tempRect.centerx = renderwidth/2
        elif alignment == 'right':
            tempRect.right = renderwidth
        topCoord += tempRect.height + spacing
        textSurf.blit(i, tempRect)

    return textSurf


# Main: Testing
if __name__ == "__main__":

    SCALE = 1.5
    def Scale(metric):
        return int(metric // SCALE)

    ScreenWidth = Scale(1280)
    ScreenHeight = Scale(960)
    TextShift = Scale(25)
    TextHeight = Scale(300)

    pygame.init()

    Screen = pygame.display.set_mode((ScreenWidth, ScreenHeight))
    Screen.fill((255, 0, 255))

    TextSurf = pygame.Surface((ScreenWidth - 2 * TextShift, TextHeight))
    TextSurf.fill((255,255,255))

    BasicFont = pygame.font.Font('assets/times.ttf', 40)

    something = TextWrap("やんでれ ² somewhere over the works, waaaaaaaaay¥nhigh¥ni like to fly in t
he sky", BasicFont, ScreenWidth - 2 * TextShift, TextShift)
    hhh = TextRender(something, BasicFont, 10, color = (0, 255, 0))
    Screen.blit(hhh, (0,0))
    pygame.display.flip()

    input()
```

39. pyTimer.py

```
"""
pyTimer.py
"""

import pygame
from math import *
from GlobalVar import *

class BgTimer:

    def __init__(self, timerLength, clock): # timerLength in seconds

        self.timerLength = timerLength # self.timerLength in seconds
        self.clock = clock
        self.Reset() # self.timerLeft in milliseconds

    def Reset(self):
        self.timeLeft = self.timerLength * 1000

    def Update(self):
        self.timeLeft -= self.clock.get_time()
        if self.timeLeft <= 0:
            self.timeLeft = 0
            return False
        else:
            return True

class Timer(BgTimer):

    def __init__(self, timerLength, clock, fontSize, font = BlockyFont, color = (0, 0, 0)):
        super().__init__(timerLength, clock)
        self.font = pygame.font.Font(font, fontSize)
        self.color = color

    def Surface(self):

        mins = floor(self.timeLeft / 1000) // 60
        sec = floor(self.timeLeft / 1000) % 60
        ms = floor((self.timeLeft // 100) % 10)
        Surf = self.font.render(f"{mins:02}:{sec:02}.{ms:01}", 1, self.color)
        return Surf

    def Time(self):

        return self.timeLeft / 1000 if self.timeLeft >= 0 else 0

def main():
    pygame.init()
    Screen = pygame.display.set_mode((900, 600))
    myClock = pygame.time.Clock()
    myTimer = Timer(3, myClock, 30,color=(100, 200, 200))

    while myTimer.Update():
        Screen.fill((0,0,0))
        Screen.blit(myTimer.Surface(), (0, 0))
        pygame.display.update()
        myClock.tick(30)

if __name__ == "__main__":
    main()
```

40. mDebug.py

```
"""
mDebug.py
"""

import json
import GlobalVar
import APlayer
from pygame.locals import *

GLOBALDEBUG = GlobalVar.SettingDict["Debug"]
EASYMODE = GlobalVar.SettingDict["EasyMode"]
DEBUGDAMAGE = 0
RAISECODE = -1

def ReloadPlayer(event):
    if GLOBALDEBUG:
        if event.type == KEYDOWN:
            if event.key == K_F12:
                playerInfo = APlayer.LoadSave(GlobalVar.PlayerDataFileName)
                return playerInfo

def Damage(dmg):

    if DEBUGDAMAGE:
        print(dmg)

def EasyMode(questionObj):

    if EASYMODE:
        return {
            "comment": "Easy Mode.",
            "diffPt": 0,
            "time": 3,
            "question": "Easy Mode: The answer is 1.",
            "answer": 1
        }
    else:
        return questionObj


def RaiseError(e, code = 0):

    if GLOBALDEBUG and RAISECODE != -1:
        if RAISECODE == 0:
            raise e
        elif code == RAISECODE:
            raise e


def SaveState(questionObj, answer, correct, timeUsed):

    if GLOBALDEBUG:

        print(correct, timeUsed, questionObj)

        try:
            file = open(GlobalVar.DebugFile, "r")
            obj = json.loads(file.read())
        except Exception as e:
            obj = {}

        try:
            obj[questionObj["comment"]].append([timeUsed, questionObj, answer, correct])
        except Exception as e:
            obj[questionObj["comment"]] = [timeUsed, questionObj, answer, correct]

        file = open(GlobalVar.DebugFile, "w")
        file.write(json.dumps(obj, indent=4))
        file.close()
```

```
def QuestionType(comment):

    if GLOBALDEBUG:
        return "(" + comment + ") "
    else:
        return ""
```

41. mErrors.py

```
"""
mErrors.py
"""

class GameError(Exception):
    pass


class QuestionError(GameError):

    def __int__(self):
        self.message = "QUestions.json not found."


class BossError(GameError):

    def __int__(self):
        self.message = "Bosses.txt not found."
```

Appendix D. Program Files

1. *files/PlayerData.txt*


```
{
    "skills": {
        "strengthPt": 0,
        "defensePt": 0,
        "timePt": 0,
        "agilityPt": 0,
        "healthPt": 0
    },
    "equipment": {
        "chestplate": 0,
        "leggings": 0,
        "sword": 0
    },
    "potions": {
        "regenPot": 0,
        "attackPot": 0,
        "defendPot": 0
    },
    "level": 1,
    "exp": 0,
    "skillPt": 0,
    "bal": 0,
    "bosslevel": 1,
    "beat": false
}
```

2. *files/Bosses.txt*


```
{
    "Bosses": [
        {
            "diff": 1,
            "name": "Archimedes",
            "fullHP": 3300,
            "fullAP": 80,
            "graphics": "assets/BossImage/Archimedes.png",
            "rewards": {
                "exp": 0,
                "cash": 0
            },
            "dialogue": [
                [
                    "Narrator",
                    "Archimedes (287-212 BC) is regarded as one of the leading scientists in classical antiquity. Besides his discovery of the Archimedes principle, a physical law that governs buoyancy forces in a fluid, he is also the first for rigorously calculating the value of pi. One of his most remarkable and famous geometric results was determining the area of a parabolic section. (We can easily determine it by integration now.)"
                ],
                [
                    "Narrator",
                    "His approach to the calculating pi was a geometrical approach using polygons. As a result, pi is sometimes referred to as ¥"Archimedes' constant¥". He estimated pi by constructing and computing the perimeters of both circumscribed and inscribed polygons, and got a result of 223/71 < pi < 22/7."
                ],
                [
                    "Archimedes",
                    "I have invented a screw for lifting water upwards! Cool, right?"
                ]
            ],
            "endDialogue": [
                [
                    "Archimedes",
                    "Oh no! You have screwed me up! Eureka! You shall be praised for your magnificent skills!"
                ]
            ]
        },
        {
            "diff": 1,
            "name": "Euclid",
            "fullHP": 7700,
            "fullAP": 120,
            "graphics": "assets/BossImage/Euclid.png",
            "rewards": {
                "exp": 30422,
                "cash": 100000
            },
            "dialogue": [
                [
                    "Narrator",
                    "Euclid (323-283 BC) is best known for his work Euclid's Elements, a 13-volume book, which contains theorems in geometry and number theory.  His contribution has formed the basis for geometry we know today. Most of the theorems we learn at school can be found in this book."
                ],
                [
                    "Narrator",
                    "Euclidean geometry, the one we are most familiar with, is based on 5 assumptions, one of which is that there are only one possible line passing through a given point such that it is parallel to a specific line. (There may be no such lines or more than one in other geometries like hyperbolic geometry and spherical geometry, grouped under non-Euclidean Geometry.)"
                ],
                [
                    "Euclid",
                    "Hello! Are you a geometry fan?"
                ]
            ],
            "endDialogue": [
                [
                    "Euclid",
                    "You don¥u2019t have to beat me up if you aren¥u2019t a fan of geometry!"
                ]
            ]
        },
        {
            "diff": 2,
            "name": "Isaac Newton",
            "fullHP": 12000,
            "fullAP": 510,
            "graphics": "assets/BossImage/IsaacNewton.png",
            "rewards": {
                "exp": 0,
                "cash": 0
```

```
            },
            "dialogue": [
                [
                    "Narrator",
                    "Isaac Newton (1642-1726/27) is widely recognized as one of the most influential scientists and Mathe
matician of all time. His book Mathematical Principles of Natural Philosophy has laid the foundations of classical mech
anics."
                ],
                [
                    "Narrator",
                    "In mathematics, he shares credit with Leibniz (Leibniz¥u2019s notation is a way of representing deri
vatives) in developing calculus, the idea of infinitesimal changes. There is also an unending list of contributions he
has made."
                ],
                [
                    "Narrator",
                    "Newton¥u2019s second law states that the total force F on an object is proportional to its mass m and
 its acceleration a. Acceleration is the rate of change of velocity v (speed with direction), and in calculus we repres
ent this relationship as a = dv/dt, where dv/dt represents the rate of change of speed v with time t."
                ],
                [
                    "Isaac Newton",
                    "I don¥u2019t like being challenged."
                ]
            ],
            "endDialogue": [
                [
                    "Issac Newton",
                    "You¥u2019re worthy of challenging me."
                ]
            ]
        },
        {
            "diff": 2,
            "name": "Carl Friedrich Gauss",
            "fullHP": 30000,
            "fullAP": 550,
            "graphics": "assets/BossImage/CarlFriedrichGauss.png",
            "rewards": {
                "exp": 170000,
                "cash": 210000
            },
            "dialogue": [
                [
                    "Narrator",
                    "Carl Friedrich Gauss (1777-1855) was a German Mathematician and physicist who made significant contr
ibutions to many fields in mathematics and science. He had extraordinary calculative powers, and has proved numerous im
portant mathematical theorem. At age 24 he published Disquisitiones Arithmeticae, one of the greatest books of pure mat
hematics ever."
                ],
                [
                    "Carl Friedrich Gauss",
                    "By the Divergence Theorem, I can the interior of you just by looking at your exterior that you are in
experienced! I¥u2019ll will prove it to you, right here, right now!"
                ]
            ],
            "endDialogue": [
                [
                    "Carl Friedrich Gauss",
                    "I shouldn¥u2019t have judge a book by its cover! You shall become a great Mathematician!"
                ]
            ]
        },
        {
            "diff": 3,
            "name": "Joseph-Louis Lagrange",
            "fullHP": 30000,
            "fullAP": 860,
            "graphics": "assets/BossImage/Lagrange.png",
            "rewards": {
                "exp": 0,
                "cash": 0
            },
            "dialogue": [
                [
                    "Narrator",
                    "Joseph-Louis Lagrange made significant contributions to the fields of analysis, number theory, and b
oth classical and celestial mechanics. He likes to use analysis as opposed to geometric methods. ¥"No diagrams will be
found in this work¥" is what he wrote in the preface to his masterpiece Analytic Mechanics."
                ]
            ],
            "endDialogue": [
                [
                    "Narrator",
                    "Lagrange once wrote ¥"As long as algebra and geometry have been separated, their progress have been s
low and their uses limited; but when these two sciences have been united, they have lent each mutual forces, and have m
```

```
arched together towards perfection.¥""
                ]
            ]
        },
        {
            "diff": 3,
            "name": "Leonhard Euler",
            "fullHP": 100000,
            "fullAP": 750,
            "graphics": "assets/BossImage/LeonhardEuler.png",
            "rewards": {
                "exp": 600000,
                "cash": 536000
            },
            "dialogue": [
                [
                    "Narrator",
                    "Euler (1707-1783) was one of the most eminent Mathematicians of the 18th century and is held to be on
e of the greatest in history. He is also widely considered to be the most prolific, as his collected works fill 92 volu
mes."
                ],
                [
                    "Narrator",
                    "He has worked on many fields in Mathematics. Due to his countless theorems, there are a lot of ambigu
ous ¥u201cEuler's equations¥u201d and ¥u201cEuler's formulas¥u201d. One of which you should know is the formula F + V -
 E = 2, which relates the number of faces, vertices and edges in a polygon."
                ],
                [
                    "Narrator",
                    "Euler¥u2019s number e is a constant which has a value of 2.718281... If you plot an exponential graph
 with it, the slope at each point of the graph is equal to its value (y-coordinate) at that point!"
                ],
                [
                    "Narrator",
                    "The most renowned and elegant identity that Euler has ever derived is the ¥u201cEuler¥u2019s identit
y¥u201d e^(i*pi)+1 = 0. This formula combines 5 of the most important constants in mathematics, including the Euler¥u20
19s number e, the imaginary number i, pi, 1 and 0."
                ]
            ],
            "endDialogue": []
        },
        {
            "diff": 4,
            "name": "Gorilla",
            "fullHP": 500,
            "fullAP": 210,
            "graphics": "assets/BossImage/Gorilla.png",
            "rewards": {
                "exp": 800,
                "cash": 3200
            },
            "dialogue": [
                [
                    "Gorilla",
                    "UGAUGAUGAUGA"
                ]
            ],
            "endDialogue": [
                [
                    "Gorilla",
                    "UGAUgaugau..."
                ]
            ]
        },
        {
            "diff": 4,
            "name": "Monkey",
            "fullHP": 700,
            "fullAP": 250,
            "graphics": "assets/BossImage/Monkey.png",
            "rewards": {
                "exp": 1000,
                "cash": 3000
            },
            "dialogue": [
                [
                    "Monkey",
                    "I'm smarter than stupid gorillas!"
                ]
            ],
            "endDialogue": [
                [
                    "Monkey",
                    "But not as smart as you..."
                ]
            ]
```

```
},
{
    "diff": 5,
    "name": "Insane Salaryman",
    "fullHP": 3500,
    "fullAP": 750,
    "graphics": "assets/BossImage/Boy.png",
    "rewards": {
        "exp": 3900,
        "cash": 6900
    },
    "dialogue": [
        [
            "Insane Salaryman",
            "I'm so insane I could do math while I beat up my boss who just fired me!!!"
        ]
    ],
    "endDialogue": [
        [
            "Insane Salaryman",
            "..."
        ]
    ]
},
{
    "diff": 5,
    "name": "Crazy Office Lady",
    "fullHP": 3200,
    "fullAP": 700,
    "graphics": "assets/BossImage/Girl.png",
    "rewards": {
        "exp": 3500,
        "cash": 7100
    },
    "dialogue": [],
    "endDialogue": []
},
{
    "diff": 6,
    "name": "A Smart Boy",
    "fullHP": 7000,
    "fullAP": 2400,
    "graphics": "assets/BossImage/SmartBoy.png",
    "rewards": {
        "exp": 9100,
        "cash": 12000
    },
    "dialogue": [
        [
            "A Smart Boy",
            "I'm smart!"
        ]
    ],
    "endDialogue": [
        [
            "A Smart Boy",
            "I'm still smart, I hope..."
        ]
    ]
},
{
    "diff": 6,
    "name": "A Smart Girl",
    "fullHP": 6500,
    "fullAP": 2000,
    "graphics": "assets/BossImage/SmartGirl.png",
    "rewards": {
        "exp": 8700,
        "cash": 11000
    },
    "dialogue": [
        [
            "A Smart Girl",
            "I am a girl who can do math!"
        ]
    ],
    "endDialogue": [
        [
            "A Smart Girl",
            "Good job!"
        ]
    ]
},
{
    "diff": 7,
    "name": "Archimedes",
```

```
    "fullHP": 4000,
    "fullAP": 13200,
    "graphics": "assets/BossImage/Archimedes.png",
    "rewards": {
        "exp": 15000,
        "cash": 13656
    },
    "dialogue": [
        [
            "Archimedes",
            "I refuse to screw up again!"
        ]
    ],
    "endDialogue": [
        [
            "Archimedes",
            "Screw it! You win!"
        ]
    ]
},
{
    "diff": 7,
    "name": "Euclid",
    "fullHP": 15000,
    "fullAP": 4500,
    "graphics": "assets/BossImage/Euclid.png",
    "rewards": {
        "exp": 15500,
        "cash": 13355
    },
    "dialogue": [
        [
            "Euclid",
            "I'll make you love geometry, and number theory while we're at it!"
        ]
    ],
    "endDialogue": [
        [
            "Euclid",
            "Stop beating me up! I'm not those annoying estate agents!"
        ]
    ]
},
{
    "diff": 7,
    "name": "Isaac Newton",
    "fullHP": 14660,
    "fullAP": 4780,
    "graphics": "assets/BossImage/IsaacNewton.png",
    "rewards": {
        "exp": 16000,
        "cash": 13000
    },
    "dialogue": [
        [
            "Isaac Newton",
            "Let's battle again and see if you're good or not."
        ]
    ],
    "endDialogue": [
        [
            "Issac Newton",
            "Bravo, bravo."
        ]
    ]
},
{
    "diff": 7,
    "name": "Carl Friedrich Gauss",
    "fullHP": 16000,
    "fullAP": 5500,
    "graphics": "assets/BossImage/CarlFriedrichGauss.png",
    "rewards": {
        "exp": 15000,
        "cash": 14500
    },
    "dialogue": [
        [
            "Carl Friedrich Gauss",
            "Oh, it's you again!"
        ]
    ],
    "endDialogue": [
        [
            "Carl Friedrich Gauss",
            "You're quite the genius!"
```

```
            ]
        ]
    },
    {
        "diff": 7,
        "name": "Joseph-Louis Lagrange",
        "fullHP": 20000,
        "fullAP": 4000,
        "graphics": "assets/BossImage/Lagrange.png",
        "rewards": {
            "exp": 18000,
            "cash": 16000
        },
        "dialogue": [],
        "endDialogue": []
    },
    {
        "diff": 7,
        "name": "Leonhard Euler",
        "fullHP": 25000,
        "fullAP": 7000,
        "graphics": "assets/BossImage/LeonhardEuler.png",
        "rewards": {
            "exp": 24000,
            "cash": 17000
        },
        "dialogue": [],
        "endDialogue": []
    },
    {
        "diff": 7,
        "name": "yandere",
        "fullHP": 32000,
        "fullAP": 9000,
        "graphics": "assets/BossImage/Yandere.png",
        "rewards": {
            "exp": 30000,
            "cash": 52000
        },
        "dialogue": [
            [
                "...",
                "suki desu yo, suugaku ga."
            ],
            [
                "...",
                "kono kamoku de watashi ga ichiban ja nai to komaru kara, koroshite morau wa."
            ]
        ],
        "endDialogue": [
            [
                "...",
                "yurusenai. zetta yurusanai."
            ]
        ]
    }
]
}
```

3. *files/Questions.json*


```
[
        {
                "comment" : "Sequence: AS Q1",
                "diff" : 2,
                "diffPt" : 0,
                "time" : 120,
                "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-300, 300, -30, 30)¥"}, ¥"int¥": {¥"objname¥":
 ¥"array¥", ¥"args¥": ¥"(¥¥¥"randint¥¥¥", 3, (2, 20))¥"}}",
                "questionStm" : "If {as,valueByNo,{int,attr,0,value}} and {as,valueByNo,{int,attr,1,value}} are th
e {int,attr,0,ord} and {int,attr,1,ord} terms of an arithmetic sequence respectively, find the {int,attr,2,ord} term o
f the sequence.",
                "answerStm" : "{as,valueByNo,{int,attr,2,value}}"
        },
        {
                "comment" : "Sequence: AS Q2",
                "diff" : 2,
                "diffPt" : 0,
                "time" : 120,
                "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-300, 300, -30, 30)¥"}, ¥"int¥": {¥"objname¥":
 ¥"array¥", ¥"args¥": ¥"(¥¥¥"randint¥¥¥", 2, (2, 20))¥"}}",
                "questionStm" : "If {as,valueByNo,{int,attr,0,value}} and {as,valueByNo,{int,attr,1,value}} are th
e {int,attr,0,ord} and {int,attr,1,ord} terms of an arithmetic sequence respectively, find the first term of the seque
nce.",
                "answerStm" : "{as,first}"
        },
        {
                "comment" : "Sequence: AS Q3",
                "diff" : 2,
                "diffPt" : 0,
                "time" : 120,
                "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-300, 300, -30, 30)¥"}, ¥"int¥": {¥"objname¥":
 ¥"array¥", ¥"args¥": ¥"(¥¥¥"randint¥¥¥", 2, (2, 20))¥"}}",
                "questionStm" : "If {as,valueByNo,{int,attr,0,value}} and {as,valueByNo,{int,attr,1,value}} are th
e {int,attr,0,ord} and {int,attr,1,ord} terms of an arithmetic sequence respectively, find the common difference of th
e sequence.",
                "answerStm" : "{as,diff}"
        },
        {
                "comment" : "Sequence: AS Q4",
                "diff" : 2,
                "diffPt" : -0.1,
                "time" : 90,
                "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-300, 300, -30, 30)¥"}, ¥"int¥": {¥"objname¥":
 ¥"randint¥", ¥"args¥": ¥"(7, 120)¥"}}",
                "questionStm" : "Find the {int,ord} term of an arithmetic sequence whose first term is {as,first} an
d common difference is {as,diff}.",
                "answerStm" : "{as,valueByNo,{int,value}}"
        },
        {
                "comment" : "Sequence: AS Q5",
                "diff" : 5,
                "diffPt" : 0.3,
                "time" : 360,
                "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-120, 120, -10, 10)¥"}, ¥"index¥": {¥"objname
¥": ¥"array¥", ¥"args¥": ¥"(¥¥¥"randint¥¥¥", 3, (2, 12))¥"}, ¥"m¥": {¥"objname¥": ¥"array¥", ¥"args¥": ¥"(¥¥¥"randint
¥¥¥", 3, (-4, 4))¥"}, ¥"x¥": {¥"objname¥": ¥"randint¥", ¥"args¥": ¥"(-10, 10)¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"
args¥": ¥"()¥"}}",
                "questionStm" : "If {m,attr,0,value,_hideone}x{eval,evalInt,{as,valueByNo,{index,sortedAttr,0,valu
e}}-({m,attr,0,value}*{x,value}),_showsign}, {m,attr,1,value,_hideone}x{eval,evalInt,{as,valueByNo,{index,sortedAttr,
1,value}}-({m,attr,1,value}*{x,value}),_showsign} and {m,attr,2,value,_hideone}x{eval,evalInt,{as,valueByNo,{index,s
ortedAttr,2,value}}-({m,attr,2,value}*{x,value}),_showsign} are the {index,sortedAttr,0,ord}, {index,sortedAttr,1,or
d} and {index,sortedAttr,2,ord} term in an arithmetic sequence respectively, find x.",
                "answerStm" : "{x,value}"
        },
        {
                "comment" : "Sequence: AS Q6",
                "diff" : 2,
                "diffPt" : 0,
                "time" : 120,
                "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-300, 300, -30, 30)¥"}, ¥"eval¥": {¥"objname
¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                "questionStm" : "Insert 3 numbers between {as,valueByNo,1} and {as,valueByNo,5} such that the 5 num
bers form an arithmetic sequence. (Seperate the numbers with commas.)",
                "answerStm" : "¥"[¥"+str({as,valueByNo,2})+¥",¥"+str({as,valueByNo,3})+¥",¥"+str({as,valueByNo,
4})+¥"]¥""
        },
        {
                "comment" : "Sequence: AS Q7",
                "diff" : 2,
                "diffPt" : 0.05,
                "time" : 180,
                "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(1, 40, 2, 10)¥"}}",
                "questionStm" : "It is given that 3 positive integers form an arithmetic sequence. Find these 3 inte
```

```
gers if their sum is {as,sumByNo,3} and product is {as,product,1,3}. (Seperate the numbers with commas.)",
                    "answerStm" : "¥"(¥"+str({as,valueByNo,1})+¥",¥"+str({as,valueByNo,2})+¥",¥"+str({as,valueByNo,
3})+¥")¥""
          },
          {
                    "comment" : "Sequence: AS Q8",
                    "diff" : 2,
                    "diffPt" : -0.1,
                    "time" : 120,
                    "obj" : "{¥"as¥": {¥"objname¥": ¥"multias¥", ¥"args¥": ¥"(5, 30)¥"}, ¥"index1¥": {¥"objname¥": ¥"ra
ndint¥", ¥"args¥": ¥"(3, 30)¥"}, ¥"index2¥": {¥"objname¥": ¥"randint¥", ¥"args¥": ¥"(60, 140)¥"}, ¥"eval¥": {¥"objname
¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "How many integers between {eval,evalInt,{as,valueByNo,{index1,value}}-{eval,evalR
and,2,{as,base}}+1} and {eval,evalInt,{as,valueByNo,{index2,value}}-{eval,evalRand,1,{as,base}}+1} exclusive are divi
sible by {as,base}?",
                    "answerStm" : "{index2,value} - {index1,value}"
          },
          {
                    "comment" : "Sequence: AS Q9",
                    "diff" : 2,
                    "diffPt" : 0,
                    "time" : 120,
                    "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-300, 300, -10, 10)¥"}, ¥"index¥": {¥"objname
¥": ¥"randint¥", ¥"args¥": ¥"(15, 70)¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "Consider the arithmetic sequence {as,valueByNo,1}, {as,valueByNo,2}, {as,valueByN
o,3}, {as,valueByNo,4}, ..., find the {as,inequality} term in the sequence such that it is {as,limit} {eval,evalInt,{a
s,valueByNo,{index,value}}+{eval,evalRand,{eval,evalInt,abs({as,diff})-1},1}*{eval,evalInt,1 if {as,diff}>1 else -
1}}.",
                    "answerStm" : "{as,valueByNo,{index,value}}"
          },
          {
                    "comment" : "Sequence: AS Q10",
                    "diff" : 5,
                    "diffPt" : 0.2,
                    "time" : 180,
                    "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-100, 100, -15, 15)¥"}, ¥"array¥": {¥"objname
¥": ¥"array¥", ¥"args¥": ¥"(¥¥¥"randint¥¥¥", 3, (4, 15))¥"}}",
                   "questionStm" : "Let S(n) denote the sum of the first n terms in an arithmetic sequence. If S({array,
sortedAttr,1,value}) = {as,sumByNo,{array,sortedAttr,1,value}}, and {as,valueByNo,{array,sortedAttr,0,value}} is the
{array,sortedAttr,0,ord}term of the sequence, find S({array,sortedAttr,2,value}).",
                    "answerStm" : "{as,sumByNo,{array,sortedAttr,2,value}}"
          },
          {
                    "comment" : "Sequence: AS Q11",
                    "diff" : 5,
                    "diffPt" : 0.2,
                    "time" : 210,
                    "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-100, 100, -15, 15)¥"}, ¥"array¥": {¥"objname
¥": ¥"array¥", ¥"args¥": ¥"(¥¥¥"randint¥¥¥", 3, (3, 20))¥"}}",
                   "questionStm" : "Let S(n) denote the sum of the first n terms in an arithmetic sequence. If S({array,
sortedAttr,0,value}) = {as,sumByNo,{array,sortedAttr,0,value}}, S({array,sortedAttr,1,value}) = {as,sumByNo,{array,so
rtedAttr,1,value}} and S(n) = {as,sumByNo,{array,sortedAttr,2,value}}, find n.",
                    "answerStm" : "{array,sortedAttr,2,value}"
          },
          {
                    "comment" : "Sequence: AS Q12",
                    "diff" : 5,
                    "diffPt" : 0.1,
                    "time" : 120,
                    "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-100, 100, -15, 15)¥"}, ¥"index¥": {¥"objname
¥": ¥"randint¥", ¥"args¥": ¥"(15, 70)¥"},¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "Consider the arithmetic sequence {as,valueByNo,1}, {as,valueByNo,2}, {as,valueByN
o,3}, {as,valueByNo,4}, ..., find the sum of all the terms that are {as,limit} {eval,evalInt,{as,valueByNo,{index,valu
e}}+{eval,evalRand,{eval,evalInt,abs({as,diff})-1},1}*{eval,evalInt,1 if {as,diff}>1 else -1}}.",
                    "answerStm" : "{as,sumByNo,{index,value}}"
          },
          {
                    "comment" : "Sequence: AS Q13",
                    "diff" : 2,
                    "diffPt" : -0.1,
                    "time" : 120,
                    "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-1000, 1000, -50, 50)¥"}, ¥"index¥": {¥"objnam
e¥": ¥"randint¥", ¥"args¥": ¥"(15, 70)¥"}}",
                    "questionStm" : "Find the sum of the first {index,value} terms of the arithmetic sequence {as,value
ByNo,1}, {as,valueByNo,2}, {as,valueByNo,3}, {as,valueByNo,4}, ...",
                    "answerStm" : "{as,sumByNo,{index,value}}"
          },
          {
                    "comment" : "Sequence: AS Q14",
                    "diff" : 2,
                    "diffPt" : 0.05,
                    "time" : 180,
                    "obj" : "{¥"as¥": {¥"objname¥": ¥"as¥", ¥"args¥": ¥"(-1000, 1000, -50, 50)¥"},  ¥"index¥": {¥"objna
me¥": ¥"randint¥", ¥"args¥": ¥"(15, 70)¥"},¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "{eval,evalInt,{index,value}-1} numbers are inserted between {as,valueByNo,1} and
{as,valueByNo,{eval,evalInt,{index,value}+1}} to form an arithmetic sequence. Find the sum of the {eval,evalInt,{index,
```

```
value}-1} numbers inserted.",
                    "answerStm" : "{as,sumByNo,{eval,evalInt,{index,value}}}-{as,valueByNo,1}"
          },
          {
                    "comment" : "Sequence: GS Q1",
                    "diff" : 3,
                    "diffPt" : 0,
                    "time" : 150,
                    "obj" : "{¥"gs¥": {¥"objname¥": ¥"gs¥", ¥"args¥": ¥"()¥"}, ¥"index¥": {¥"objname¥": ¥"array¥", ¥"ar
gs¥": ¥"(¥¥¥"randint¥¥¥", 3, (2, 8))¥"},¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "If the {index,attr,0,ord} and {index,attr,1,ord} term of a geometric sequence are
{eval,evalFraction,{gs,valueByNo,{index,attr,0,value}}} and {eval,evalFraction,{gs,valueByNo,{index,attr,1,value}}}
respectively, find the {index,attr,2,ord} term of the sequence, given that the common ratio is {gs,sign}.",
                    "answerStm" : "{eval,evalFraction,{gs,valueByNo,{index,attr,2,value}}}"
          },
          {
                    "comment" : "Sequence: GS Q2",
                    "diff" : 6,
                    "diffPt" : 0.25,
                    "time" : 240,
                    "obj" : "{¥"gs¥": {¥"objname¥": ¥"gs¥", ¥"args¥": ¥"(¥¥¥"int¥¥¥")¥"}, ¥"m¥": {¥"objname¥": ¥"array
¥", ¥"args¥": ¥"(¥¥¥"randint¥¥¥", 3, (-3, 3))¥"}, ¥"x¥": {¥"objname¥": ¥"randint¥", ¥"args¥": ¥"(-7, 7)¥"}, ¥"eval¥":
{¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}¥n",
                    "questionStm" : "If {m,attr,0,value,_hideone}x{eval,evalFraction,{gs,valueByNo,1}-{m,attr,0,value}
*{x,value},20,20,_showsign}, {m,attr,0,value,_hideone}x{eval,evalFraction,{gs,valueByNo,2}-{m,attr,0,value}*{x,valu
e},20,20,_showsign} and {m,attr,0,value,_hideone}x{eval,evalFraction,{gs,valueByNo,3}-{m,attr,0,value}*{x,value},20,
20,_showsign} form a geometric sequence, find x given that the common ratio is {gs,sign} and that x is an integer.",
                    "answerStm" : "{x,value}"
          },
          {
                    "comment" : "Sequence: GS Q3",
                    "diff" : 3,
                    "diffPt" : 0.1,
                    "time" : 180,
                    "obj" : "{¥"gs¥": {¥"objname¥": ¥"gs¥", ¥"args¥": ¥"()¥"}, ¥"x¥": {¥"objname¥": ¥"fraction¥", ¥"arg
s¥": ¥"(3, -3, 2, 7)¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "Suppose {eval,varDisplay,({gs,valueByNo,3})/({x,value})}, {eval,evalFraction,{gs,
valueByNo,4}} and {eval,varDisplay,({gs,valueByNo,5})/({x,value})} form a geometric sequence. Find x given that the co
mmon ratio is {gs,sign}.",
                    "answerStm" : "{eval,evalFraction,{x,value}}"
          },
          {
                    "comment" : "Sequence: GS Q4",
                    "diff" : 3,
                    "diffPt" : 0,
                    "time" : 120,
                    "obj" : "{¥"gs¥": {¥"objname¥": ¥"gs¥", ¥"args¥": ¥"(¥¥¥"dec¥¥¥")¥"}, ¥"eval¥": {¥"objname¥": ¥"eva
l¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "Find the sum to inifinity of the following geometric sequence: {eval,evalFraction,
{gs,valueByNo,1}}, {eval,evalFraction,{gs,valueByNo,2}}, {eval,evalFraction,{gs,valueByNo,3}}, {eval,evalFraction,{g
s,valueByNo,4}}, ...",
                    "answerStm" : "{eval,evalFraction,{gs,sumByNo,inf}}"
          },
          {
                    "comment" : "Sequence: GS Q5",
                    "diff" : 6,
                    "diffPt" : 0.3,
                    "time" : 270,
                    "obj" : "{¥"gs¥": {¥"objname¥": ¥"gs¥", ¥"args¥": ¥"(¥¥¥"dec¥¥¥")¥"}, ¥"eval¥": {¥"objname¥": ¥"eva
l¥", ¥"args¥": ¥"()¥"}, ¥"int¥": {¥"objname¥": ¥"randint¥", ¥"args¥": ¥"(3,8)¥"}, ¥"rand¥": {¥"objname¥": ¥"randint¥",
 ¥"args¥": ¥"(3,8)¥"}}",
                    "questionStm" : "Let T(n) denote the nth term in a geometric sequence. If the sum to infinity T(1) +
 T(2) + T(3) + ... is {eval,evalFraction,{gs,sumByNo,inf},100000,100} and the sum of its {eval,eval,¥"odd¥" if {int,va
lue}%2 == 1 else ¥"even¥"} numbered terms {eval,eval,¥"T(1) + T(3) + T(5) + ...¥" if {int,value}%2 == 1 else ¥"T(2) + T
(4) + T(6) + ...¥"} is {eval,evalFraction,({gs,sumOdd,inf}) if {int,value}%2 == 1 else ({gs,sumByNo,inf}-{gs,sumOdd,in
f}),100000,100}, find T({rand,value}) given that the common ratio is {gs,sign}.",
                    "answerStm" : "{eval,evalFraction,{gs,valueByNo,{rand,value}},100000,50}"
          },
          {
                    "comment" : "Quadratic Equation: Q1",
                    "diff" : 1,
                    "diffPt" : -0.1,
                    "time" : 60,
                    "obj" : "{¥"q¥": {¥"objname¥": ¥"quadratic¥", ¥"args¥": ¥"()¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥",
¥"args¥": ¥"()¥"}}",
                    "questionStm" : "Solve {eval,varDisplay,{q,displayCoeff,2},2,False,True} {eval,varDisplay,{q,displ
ayCoeff,1},1,True,True} {eval,varDisplay,{q,displayCoeff,0},0,True,True} = 0 for x. (Seperate your answers with comma
s).",
                    "answerStm" : "{eval,eval,{q,ansRoots}}"
          },
          {
                    "comment" : "Quadratic Equation: Q2",
                    "diff" : 1,
                    "diffPt" : 0,
                    "time" : 90,
                    "obj" : "{¥"q¥": {¥"objname¥": ¥"quadratic¥", ¥"args¥": ¥"(FracValue(-1.5, 1.5, -5, 5))¥"}, ¥"eval
```

```
¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "Solve {eval,varDisplay,{q,displayCoeff,2},2,False,True} {eval,varDisplay,{q,displ
ayCoeff,1},1,True,True} {eval,varDisplay,{q,displayCoeff,0},0,True,True} = 0 for x. (Seperate your answers with comma
s).",
                    "answerStm" : "{eval,eval,{q,ansRoots}}"
          },
          {
                    "comment" : "Quadratic Equation: Q3",
                    "diff" : 1,
                    "diffPt" : 0,
                    "time" : 180,
                    "obj" : "{¥"q¥": {¥"objname¥": ¥"quadratic¥", ¥"args¥": ¥"(random.randint(-5, 5))¥"}, ¥"eval¥": {¥"
objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}, ¥"int¥": {¥"objname¥": ¥"randint¥", ¥"args¥": ¥"(-5, 5)¥"}, ¥"frac¥": {¥"objna
me¥": ¥"fraction¥", ¥"args¥": ¥"(-5, 5, -7, 7)¥"}}",
                    "questionStm" : "Solve {eval,varDisplay,{q,displayCoeff,2},2,False,True,({eval,varDisplay,{frac,va
lue},1}){eval,varDisplay,{int,value},0,True}} {eval,varDisplay,{q,displayCoeff,1},1,True,True,({eval,varDisplay,{fra
c,value},1}){eval,varDisplay,{int,value},0,True}} = {eval,varDisplay,{eval,eval,{q,displayCoeff,0}*-1},0,False,True}
 for x.",
                    "answerStm" : "(¥"{eval,evalFraction,({q,root,0}-{int,value})/({frac,value}),30,30}¥",¥"{eval,eva
lFraction,({q,root,1}-{int,value})/({frac,value}),30,30}¥")"
          },
          {
                    "comment" : "Quadratic Equation: Q4",
                    "diff" : 4,
                    "diffPt" : 0.2,
                    "time" : 270,
                    "obj" : "{¥"int¥": {¥"objname¥": ¥"array¥", ¥"args¥": ¥"(¥¥¥"randint¥¥¥", 2, (10, 70))¥"},  ¥"eval
¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "A {eval,evalRandStr,(ladder,book,wooden board,ruler)} that is {int,sortedAttr,1,v
alue} cm in length is leaning on a vertical wall. A blow of the wind, and it has sled x cm down such that the upper end
 is now {int,sortedAttr,0,value} cm above the horizontal ground. If the bottom end has also moved x cm horizontally, fin
d x. (Correct your answer to 3 significant figures.)",
                    "answerStm" : "{eval,validate,({int,sortedAttr,1,value}**2 - {int,sortedAttr,0,value}**2)**0.5-{in
t,sortedAttr,0,value}>1} ({int,sortedAttr,1,value}**2 - {int,sortedAttr,0,value}**2)**0.5-{int,sortedAttr,0,value}"
          },
          {
                    "comment" : "Quadratic Equation: Q5",
                    "diff" : 1,
                    "diffPt" : 0,
                    "time" : 120,
                    "obj" : "{¥"q¥": {¥"objname¥": ¥"quadratic¥", ¥"args¥": ¥"(-5, roots = ( FracValue(-10, 0, 0, 5), F
racValue(0, 10, 0, 5)))¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "A ball is thrown in a way such that its altitude y m is a function of time t s descr
ibed by y(t) = {eval,varDisplay,{q,displayCoeff,0},0,False,True,t} {eval,varDisplay,{q,displayCoeff,1},1,True,True,t}
 {eval,varDisplay,{q,displayCoeff,2},2,True,True,t}. Find the time elasped (drop the unit) after the throw when the ba
ll returns to its original position, i.e. y = 0.",
                    "answerStm" : "{eval,evalFraction,{q,root,1}}"
          },
          {
                    "comment" : "Quadratic Equation: Q6",
                    "diff" : 4,
                    "diffPt" : 0.2,
                    "time" : 240,
                    "obj" : "{¥"q¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args¥": ¥"(genMode = ¥¥¥"int¥¥¥")¥"}, ¥"int¥": {¥"o
bjname¥": ¥"array¥", ¥"args¥": ¥"(¥¥¥"randint¥¥¥", 3, (-8, 8))¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"},
 ¥"frac¥": {¥"objname¥": ¥"randint¥", ¥"args¥": ¥"(-7, 7)¥"}}",
                    "questionStm" : "Suppose Q(x) is a quadratic function. If Q({int,attr,0,value}) = {eval,evalFractio
n,{q,value,{int,attr,0,value}}-{frac,value}*{int,attr,0,value},100,50}, Q({int,attr,1,value}) = {eval,evalFraction,
{q,value,{int,attr,1,value}}-{frac,value}*{int,attr,1,value},100,50} and Q({int,attr,2,value}) = {eval,evalFraction,
{q,value,{int,attr,2,value}}-{frac,value}*{int,attr,2,value},100,50}, solve Q(x) = {eval,varDisplay,-{frac,value},
1}}.",
                    "answerStm" : "{eval,eval,{q,ansRoots}}"
          },
          {
                    "comment" : "Quadratic Equation: Q7",
                    "diff" : 1,
                    "diffPt" : 0,
                    "time" : 150,
                    "obj" : "{¥"q1¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args¥": ¥"()¥"}, ¥"q2¥": {¥"objname¥": ¥"QUADRATIC
¥", ¥"args¥": ¥"()¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "Solve {eval,varDisplay,{q1,displayCoeff,2}+{q2,displayCoeff,2},2,False,True} {eva
l,varDisplay,{q1,displayCoeff,1}+{q2,displayCoeff,1},1,True,True} {eval,varDisplay,{q1,displayCoeff,0}+{q2,displayCo
eff,0},0,True,True} = ({eval,varDisplay,{q2,denon,0},1,False,True} {eval,varDisplay,-{q2,numer,0},0,True,True})({eva
l,varDisplay,{q2,denon,1},1,False,True} {eval,varDisplay,-{q2,numer,1},0,True,True}). (Seperate your answers with com
mas.)",
                    "answerStm" : "{eval,eval,{q1,ansRoots}}"
          },
          {
                    "comment" : "Quadratic Equation: Q8",
                    "diff" : 1,
                    "diffPt" : -0.1,
                    "time" : 90,
                    "obj" : "{¥"q¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args¥": ¥"()¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥",
¥"args¥": ¥"()¥"}}",
                    "questionStm" : "Find the discriminant of the quadratic equation {eval,varDisplay,{q,displayCoeff,
2},2,False,True} {eval,varDisplay,{q,displayCoeff,1},1,True,True} {eval,varDisplay,{q,displayCoeff,0},0,True,True} =
```

```
0. (Do not simplify the coefficients.)",
                    "answerStm" : "{eval,evalFraction,{q,discriminant}}"
          },
          {
                    "comment" : "Quadratic Equation: Q9",
                    "diff" : 1,
                    "diffPt" : -0.3,
                    "time" : 90,
                    "obj" : "{¥"q¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args¥": ¥"(FracValue(-10, 10, -3, 3))¥"}, ¥"eval¥":
 {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "If x = a is the symmetry axis of the quadratic graph y = {eval,varDisplay,{q,displa
yCoeff,2},2,False,True} {eval,varDisplay,{q,displayCoeff,1},1,True,True} {eval,varDisplay,{q,displayCoeff,0},0,True,
True}, find a.",
                    "answerStm" : "{eval,evalFraction,{q,symmetryAxis}}"
          },
          {
                    "comment" : "Quadratic Equation: Q10",
                    "diff" : 1,
                    "diffPt" : -0.3,
                    "time" : 90,
                    "obj" : "{¥"q¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args¥": ¥"(FracValue(-10, 10, -3, 3))¥"}, ¥"eval¥":
 {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "Find the {q,limit} value of the quadratic graph y = {eval,varDisplay,{q,displayCoe
ff,0},0,False,True,x,100,20} {eval,varDisplay,{q,displayCoeff,1},1,True,True,x,100,20} {eval,varDisplay,{q,displayCo
eff,2},2,True,True,x,100,20}.",
                    "answerStm" : "{eval,evalFraction,{q,vertex}, 100, 100}"
          },
          {
                    "comment" : "Quadratic Equation: Q11",
                    "diff" : 1,
                    "diffPt" : 0.05,
                    "time" : 120,
                    "obj" : "{¥"q¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args¥": ¥"(RandomInt(-10, 10))¥"}, ¥"eval¥": {¥"obj
name¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                    "questionStm" : "Suppose the quadratic graph {eval,varDisplay,{q,displayCoeff,2},2,False,True} {ev
al,varDisplay,{q,displayCoeff,0},0,True,True} {eval,varDisplay,{q,displayCoeff,1},1,True,True} cuts the x-axis at two
 points A and B. Let C be the vertex of the graph. Find the area of triangle ABC.",
                    "answerStm" : "{eval,validate,{q,root,1}!={q,root,0}} {eval,validate,{q,vertex}!=0} {eval,evalFrac
tion,abs({q,vertex}*({q,root,1}-{q,root,0}))/2, 100, 100}"
          },
          {
                    "comment" : "Quadratic Equation: Q12",
                    "diff" : 4,
                    "diffPt" : 0,
                    "time" : 120,
                    "obj" : "{¥"frac¥": {¥"objname¥": ¥"array¥", ¥"args¥": ¥"(¥¥¥"fraction¥¥¥", 2, (-4, 4, -7, 7))¥"},
¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                   "questionStm" : "Suppose a and b are the two solutions to the quadratic equation {eval,varDisplay,1,
2,False,True} {eval,varDisplay,{frac,attr,1,value},1,True,True} {eval,varDisplay,{frac,attr,0,value},0,True,True} =
0. Find |a^2 - b^2|, where |x| is the value of x ignoring its sign. (Correct your answer to 3 significant figures.)",
                    "answerStm" : "{eval,validate,(({frac,attr,1,value}) ** 2 - 4* {frac,attr,0,value})>0}{eval,eval,a
bs(-({frac,attr,1,value}) * (({frac,attr,1,value}) ** 2 - 4*({frac,attr,0,value}))**0.5)}"
          },
          {
                    "comment" : "Quadratic Equation: Q13",
                    "diff" : 4,
                    "diffPt" : 0.1,
                    "time" : 180,
                    "obj" : "{¥"frac¥": {¥"objname¥": ¥"array¥", ¥"args¥": ¥"(¥¥¥"fraction¥¥¥", 2, (-4, 4, -7, 7))¥"},
¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                   "questionStm" : "Suppose a and b are the two solutions to the quadratic equation {eval,varDisplay,1,
2,False,True} {eval,varDisplay,{frac,attr,1,value},1,True,True} {eval,varDisplay,{frac,attr,0,value},0,True,True}. F
ind |a^3 - b^3|,  where |x| is the value of x ignoring its sign. (Correct your answer to 3 significant figures.)",
                    "answerStm" : "{eval,validate,  ( ({frac,attr,1,value}) ** 2 - 4* {frac,attr,0,value} )  >  0}{eval,
eval,abs( ((({frac,attr,1,value}) ** 2 - 4 * ({frac,attr,0,value}))**0.5) * (({frac,attr,1,value}) ** 2 - {frac,attr,0,
value}))}"
          },
          {
                    "comment" : "Polynomial: Q1",
                    "diff" : 3,
                    "diffPt" : 0,
                    "time" : 180,
                   "obj" : "{¥"p¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(3)¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥",
 ¥"args¥": ¥"()¥"}, ¥"order¥": {¥"objname¥": ¥"array¥", ¥"args¥": ¥"(¥¥¥"randint¥¥¥", 4, (0, 3))¥"}}",
                    "questionStm" : "It is given that {p,denon,0,_hideone}x {eval,evalInt,-({p,numer,0}),_showsignspac
e} is a root of the cubic equation {eval,varDisplay,{p,displayCoeff,{order,attr,0,value}},{order,attr,0,value},False,
True} {eval,varDisplay,{p,displayCoeff,{order,attr,1,value}},{order,attr,1,value},True,True} {eval,varDisplay,{p,dis
playCoeff,{order,attr,2,value}},{order,attr,2,value},True,True} {eval,varDisplay,{p,displayCoeff,{order,attr,3,valu
e}},{order,attr,3,value},True,True} = 0. Solve the equation.",
                    "answerStm" : "{p,ansRoots}"
          },
          {
                    "comment" : "Polynomial: Q2",
                    "diff" : 3,
                    "diffPt" : 0.1,
                    "time" : 270,
```

```
                "obj" : "{¥"p¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(4, intSpec = (-5, 5), fractionSpec = (-
2, 2, -3, 3))¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}, ¥"order¥": {¥"objname¥": ¥"array¥", ¥"args¥":
¥"(¥¥¥"randint¥¥¥", 5, (0, 4))¥"}}",
                "questionStm" : "It is given that {p,denon,0,_hideone}x {eval,evalInt,-({p,numer,0}),_showsignspac
e} and {p,denon,1,_hideone}x {eval,evalInt,-({p,numer,1}),_showsignspace} are roots of the quartic equation {eval,var
Display,{p,displayCoeff,{order,attr,0,value}},{order,attr,0,value},False,True} {eval,varDisplay,{p,displayCoeff,{ord
er,attr,1,value}},{order,attr,1,value},True,True} {eval,varDisplay,{p,displayCoeff,{order,attr,2,value}},{order,attr,
2,value},True,True} {eval,varDisplay,{p,displayCoeff,{order,attr,3,value}},{order,attr,3,value},True,True} {eval,var
Display,{p,displayCoeff,{order,attr,4,value}},{order,attr,4,value},True,True} = 0. Solve the equation.",
                "answerStm" : "{eval,validate,abs({p,displayCoeff,4})<6} {p,ansRoots}"
        },
        {
                "comment" : "Polynomial: Q3",
                "diff" : 6,
                "diffPt" : 0.2,
                "time" : 270,
                "obj" : "{¥"p1¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(3, roots = randList[0:3])¥"}, ¥"p2¥":
{¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(3, roots = randList[1:4])¥"}, ¥"q¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args
¥": ¥"( roots = randIntList[1:3])¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}, ¥"order¥": {¥"objname¥": ¥"
array¥", ¥"args¥": ¥"(¥¥¥"randint¥¥¥", 4, (0, 3))¥"}}",
                "questionStm" : "Given that A{eval,varDisplay,1,2,False,False} + B{eval,varDisplay,1,1,False,Fals
e} + C is a common factor of {eval,varDisplay,{p1,displayCoeff,{order,attr,0,value}},{order,attr,0,value},False,True,
x,30} {eval,varDisplay,{p1,displayCoeff,{order,attr,1,value}},{order,attr,1,value},True,True,x,30} {eval,varDisplay,
{p1,displayCoeff,{order,attr,2,value}},{order,attr,2,value},True,True,x,30} {eval,varDisplay,{p1,displayCoeff,{order,
attr,3,value}},{order,attr,3,value},True,True,x,30} and {eval,varDisplay,{p2,displayCoeff,3},3,False,True} {eval,var
Display,{p2,displayCoeff,2},2,True,True,x,30} {eval,varDisplay,{p2,displayCoeff,1},1,True,True,x,30} {eval,varDispla
y,{p2,displayCoeff,0},0,True,True,x,30}. Find A + B + C.",
                "answerStm" : "{eval,evalFraction,{q,displayCoeff,2}+{q,displayCoeff,1}+{q,displayCoeff,0}}"
        },
        {
                "comment" : "Polynomial: Q4",
                "diff" : 6,
                "diffPt" : 0.1,
                "time" : 240,
                "obj" : "{¥"p¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(3)¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥",
 ¥"args¥": ¥"()¥"}}",
                "questionStm" : "It is given that {p,denon,0,_hideone}x {eval,evalInt,-({p,numer,0}),_showsignspac
e} is a root of the cubic polynomial P(x) = {eval,varDisplay,{p,displayCoeff,3},3,False,True} {eval,varDisplay,{p,disp
layCoeff,2},2,True,True} {eval,varDisplay,{p,displayCoeff,1},1,True,True} + p. Solve P(x) = 0.",
                "answerStm" : "{p,ansRoots}"
        },
        {
                "comment" : "Polynomial: Q5",
                "diff" : 3,
                "diffPt" : 0.2,
                "time" : 240,
                "obj" : "{¥"p¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(3)¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥",
 ¥"args¥": ¥"()¥"}, ¥"a¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(1)¥"}}",
                "questionStm" : "It is given that {p,denon,0,_hideone}x {eval,evalInt,-({p,numer,0}),_showsignspac
e} is a root of the cubic polynomial P(x) = {eval,varDisplay,{p,displayCoeff,3},3,False,True,x,5} + b{eval,varDisplay,
1,2,False,False} {eval,varDisplay,{p,displayCoeff,1},1,True,True} + a. When P(x) is divided by {eval,varDisplay,{a,di
splayCoeff,1},1,False,True} {eval,varDisplay,{a,displayCoeff,0},0,True,True}, the remainder is {p,value,{a,root,0}}.
Solve P(x) = 0.",
                "answerStm" : "{eval,validate,{eval,evalFraction,{p,value,{a,root,0}},60,30}< 60}{eval,validate,ab
s({p,root,0}) != abs({a,root,0})}{p,ansRoots}"
        },
        {
                "comment" : "Polynomial: Q6",
                "diff" : 3,
                "diffPt" : 0.1,
                "time" : 240,
                "obj" : "{¥"p¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(3, roots=randList[:3])¥"}, ¥"eval¥":
{¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}, ¥"a¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(1, roots=(randList[3],))
¥"}}",
                "questionStm" : "It is given that {p,denon,0,_hideone}x {eval,evalInt,-({p,numer,0}),_showsignspac
e} is a root of the cubic polynomial P(x) = {eval,varDisplay,{p,displayCoeff,3},3,False,True,x,5} + b{eval,varDisplay,
1,2,False,False} {eval,varDisplay,{p,displayCoeff,1},1,True,True} {eval,varDisplay,{p,displayCoeff,0},0,True,True}.
When P(x) is divided by {eval,varDisplay,{a,displayCoeff,1},1,False,True} {eval,varDisplay,{a,displayCoeff,0},0,True,
True}, the remainder is r. Find r.",
                "answerStm" : "{eval,evalFraction,{p,value,{a,root,0}},50,30}"
        },
        {
                "comment" : "Polynomial: Q7",
                "diff" : 3,
                "diffPt" : 0.1,
                "time" : 270,
                "obj" : "{¥"p1¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(3,  roots = randIntList[0:3])¥"}, ¥"ev
al¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}, ¥"p2¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(3, roots = randInt
List[2:5])¥"}}",
                "questionStm" : "If H(x) is the H.C.F. of {eval,varDisplay,{p1,displayCoeff,3},3,False,True,x,5} {e
val,varDisplay,{p1,displayCoeff,2},2,True,True} {eval,varDisplay,{p1,displayCoeff,1},1,True,True} {eval,varDisplay,
{p1,displayCoeff,0},0,True,True} and {eval,varDisplay,{p2,displayCoeff,3},3,False,True,x,5} {eval,varDisplay,{p2,dis
playCoeff,2},2,True,True} {eval,varDisplay,{p2,displayCoeff,1},1,True,True} {eval,varDisplay,{p2,displayCoeff,0},0,T
rue,True}, solve H(x) = 0.",
                "answerStm" : "{eval,eval,set({p1,ansRoots}) & set({p2,ansRoots})}"
        },
```

```
        {
                "comment" : "Polynomial: Q8",
                "diff" : 3,
                "diffPt" : 0.1,
                "time" : 270,
                "obj" : "{¥"p1¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(3, roots = randIntList[0:3])¥"}, ¥"eva
l¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}, ¥"p2¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(3, roots = randIntL
ist[2:5])¥"}}",
                "questionStm" : "If L(x) is the L.C.M. of {eval,varDisplay,{p1,displayCoeff,3},3,False,True,x,5} {e
val,varDisplay,{p1,displayCoeff,2},2,True,True} {eval,varDisplay,{p1,displayCoeff,1},1,True,True} {eval,varDisplay,
{p1,displayCoeff,0},0,True,True} and {eval,varDisplay,{p2,displayCoeff,3},3,False,True,x,5} {eval,varDisplay,{p2,dis
playCoeff,2},2,True,True} {eval,varDisplay,{p2,displayCoeff,1},1,True,True} {eval,varDisplay,{p2,displayCoeff,0},0,T
rue,True}, solve L(x) = 0.",
                "answerStm" : "{eval,eval,set({p1,ansRoots}) | set({p2,ansRoots})}"
        },
        {
                "comment" : "Polynomial: Q9",
                "diff" : 6,
                "diffPt" : 0.1,
                "time" : 360,
                "obj" : "{¥"q1¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args¥": ¥"(roots = (randList[0], randList[1]))¥"},
 ¥"q2¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args¥": ¥"(roots = (randList[1], randList[2]))¥"}, ¥"q3¥": {¥"objname¥": ¥"QUA
DRATIC¥", ¥"args¥": ¥"(roots = (randList[3], randList[2]))¥"}, ¥"linear¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"
(1, roots = (randList[3]))¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                "questionStm" : "Solve ({eval,varDisplay,{q1,displayCoeff,2},2,False,True} {eval,varDisplay,{q1,di
splayCoeff,1},1,True,True} {eval,varDisplay,{q1,displayCoeff,0},0,True,True})/({eval,varDisplay,{q2,displayCoeff,2},
2,False,True} {eval,varDisplay,{q2,displayCoeff,1},1,True,True} {eval,varDisplay,{q2,displayCoeff,0},0,True,True}) *
 ({eval,varDisplay,{q3,displayCoeff,2},2,False,True} {eval,varDisplay,{q3,displayCoeff,1},1,True,True} {eval,varDisp
lay,{q3,displayCoeff,0},0,True,True})/({eval,varDisplay,{linear,displayCoeff,1},1,False,True} {eval,varDisplay,{line
ar,displayCoeff,0},0,True,True}) = 0.",
                "answerStm" : "randList[0]"
        },
        {
                "comment" : "Polynomial: Q10",
                "diff" : 3,
                "diffPt" : 0.2,
                "time" : 420,
                "obj" : "{¥"q1¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args¥": ¥"(roots = (randList[3], randList[1]))¥"},
 ¥"q2¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(3, roots = (randList[1], randList[2], randList[4]))¥"}, ¥"q3¥":
{¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(3, roots = (randList[0], randList[2], randList[4]))¥"}, ¥"linear¥": {¥"obj
name¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(1, roots = (randList[3]))¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()
¥"}}",
                "questionStm" : "Solve ({eval,varDisplay,{q1,displayCoeff,2},2,False,True} {eval,varDisplay,{q1,di
splayCoeff,1},1,True,True} {eval,varDisplay,{q1,displayCoeff,0},0,True,True})/({eval,varDisplay,{q2,displayCoeff,3},
3,False,True,x,5} {eval,varDisplay,{q2,displayCoeff,2},2,True,True} {eval,varDisplay,{q2,displayCoeff,1},1,True,Tru
e} {eval,varDisplay,{q2,displayCoeff,0},0,True,True}) * ({eval,varDisplay,{q3,displayCoeff,3},3,False,True,x,5} {eva
l,varDisplay,{q3,displayCoeff,2},2,True,True} {eval,varDisplay,{q3,displayCoeff,1},1,True,True} {eval,varDisplay,{q3,
displayCoeff,0},0,True,True})/({eval,varDisplay,{linear,displayCoeff,1},1,False,True} {eval,varDisplay,{linear,displ
ayCoeff,0},0,True,True}) = 0.",
                "answerStm" : "randList[0]"
        },
        {
                "comment" : "Polynomial: Q11",
                "diff" : 5,
                "diffPt" : 0.1,
                "time" : 300,
                "obj" : "{¥"q1¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args¥": ¥"(roots = (randList[0], randList[1]))¥"},
 ¥"q2¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args¥": ¥"(roots = (randList[1], randList[2]))¥"}, ¥"q3¥": {¥"objname¥": ¥"QUA
DRATIC¥", ¥"args¥": ¥"(roots = (randList[3], randList[2]))¥"}, ¥"linear¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"
(1, roots = (randList[3]))¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                "questionStm" : "If [({eval,varDisplay,{q3,displayCoeff,2},2,False,True} {eval,varDisplay,{q3,disp
layCoeff,1},1,True,True} {eval,varDisplay,{q3,displayCoeff,0},0,True,True})/({eval,varDisplay,{q2,displayCoeff,2},2,
False,True} {eval,varDisplay,{q2,displayCoeff,1},1,True,True} {eval,varDisplay,{q2,displayCoeff,0},0,True,True})] /
[({eval,varDisplay,{linear,displayCoeff,1},1,False,True} {eval,varDisplay,{linear,displayCoeff,0},0,True,True})/({ev
al,varDisplay,{q1,displayCoeff,2},2,False,True} {eval,varDisplay,{q1,displayCoeff,1},1,True,True} {eval,varDisplay,
{q1,displayCoeff,0},0,True,True})] = k(x - a), where k is some abritary constant, find a.",
                "answerStm" : "randList[0]"
        },
        {
                "comment" : "Polynomial: Q12",
                "diff" : 3,
                "diffPt" : 0.2,
                "time" : 420,
                "obj" : "{¥"u1¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(1, roots = (randList[1], ))¥"}, ¥"u2
¥": {¥"objname¥": ¥"QUADRATIC¥", ¥"args¥": ¥"(roots = (randList[0], randList[2]))¥"}, ¥"u3¥": {¥"objname¥": ¥"POLYNOM
IAL¥", ¥"args¥": ¥"(3, roots = (randList[5], randList[4], randList[3]))¥"}, ¥"l1¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"ar
gs¥": ¥"(1, roots = (randList[3], ))¥"}, ¥"l2¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(2, roots = (randList[4], r
andList[5]))¥"}, ¥"l3¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(2, roots = (randList[1], randList[2]))¥"}, ¥"eval
¥": {¥"objname¥": ¥"eval¥", ¥"args¥": ¥"()¥"}}",
                "questionStm" : "Solve ({eval,varDisplay,{u1,displayCoeff,1},1,False,True} {eval,varDisplay,{u1,di
splayCoeff,0},0,True,True})/({eval,varDisplay,{l1,displayCoeff,1},1,False,True} {eval,varDisplay,{l1,displayCoeff,0},
0,True,True}) * ({eval,varDisplay,{u2,displayCoeff,2},2,False,True} {eval,varDisplay,{u2,displayCoeff,1},1,True,Tru
e} {eval,varDisplay,{u2,displayCoeff,0},0,True,True})/({eval,varDisplay,{l2,displayCoeff,2},2,False,True} {eval,varD
isplay,{l2,displayCoeff,1},1,True,True} {eval,varDisplay,{l2,displayCoeff,0},0,True,True}) / [({eval,varDisplay,{l3,
displayCoeff,2},2,False,True} {eval,varDisplay,{l3,displayCoeff,1},1,True,True} {eval,varDisplay,{l3,displayCoeff,0},
0,True,True})/({eval,varDisplay,{u3,displayCoeff,3},3,False,True} {eval,varDisplay,{u3,displayCoeff,2},2,True,True}
```

```
{eval,varDisplay,{u3,displayCoeff,1},1,True,True} {eval,varDisplay,{u3,displayCoeff,0},0,True,True})] = 0.",
                    "answerStm" : "randList[0]"
          },
          {
                    "comment" : "Polynomial: Q13",
                    "diff" : 5,
                    "diffPt" : 0.04,
                    "time" : 180,
                    "obj" : "{¥"l1¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(1, roots = (randList[0], ))¥"}, ¥"l2
¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(1, roots = (randList[1], ))¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"arg
s¥": ¥"()¥"}}",
                    "questionStm" : "Solve 1/({eval,varDisplay,{l1,displayCoeff,1},1,False,True} {eval,varDisplay,{l1,
displayCoeff,0},0,True,True}) + 2/({eval,varDisplay,{l2,displayCoeff,1},1,False,True} {eval,varDisplay,{l2,displayCo
eff,0},0,True,True}) = 0.",
                    "answerStm" : "{eval,validate,(2*({l1,numer,0})+({l2,numer,0}))/(2*({l1,denon,0})+({l2,denon,0}))
not in [{l1,root,0}, {l2,root,0}]}{eval,evalFraction,(2*({l1,numer,0})+({l2,numer,0}))/(2*({l1,denon,0})+({l2,denon,
0}))}"
          },
          {
                    "comment" : "Polynomial: Q14",
                    "diff" : 3,
                    "diffPt" : 0.04,
                    "time" : 180,
                    "obj" : "{¥"l1¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(1, roots = (randList[0], ))¥"}, ¥"l2
¥": {¥"objname¥": ¥"POLYNOMIAL¥", ¥"args¥": ¥"(1, roots = (randList[1], ))¥"}, ¥"eval¥": {¥"objname¥": ¥"eval¥", ¥"arg
s¥": ¥"()¥"}}",
                    "questionStm" : "Solve 1/({eval,varDisplay,{l1,displayCoeff,1},1,False,True} {eval,varDisplay,{l1,
displayCoeff,0},0,True,True}) - 2/({eval,varDisplay,{l2,displayCoeff,1},1,False,True} {eval,varDisplay,{l2,displayCo
eff,0},0,True,True}) = 0.",
                    "answerStm" : "{eval,validate,(2*({l1,numer,0})-({l2,numer,0}))/(2*({l1,denon,0})-({l2,denon,0}))
not in [{l1,root,0}, {l2,root,0}]}{eval,evalFraction,(2*({l1,numer,0})-({l2,numer,0}))/(2*({l1,denon,0})-({l2,denon,
0}))}"
          }
]
```

Appendix E. Sources

Creating this game would not have been possible without the resources from the following websites:

A. Graphics

1. いらすとや: *https://www.irasutoya.com/*
2. *https://www.kindpng.com/imgv/mTRhbi_freetoedit-justgirlythings-hd-png-download/*
3. Wikipedia
4. Britannica

B. Fonts

1. *https://www.dafont.com/minecraft.font*
   Minecraft by Craftron Gaming

2. *https://www.fontspace.com/*

C. Sound Effects

1. *http://dig.ccmixter.org/*

   - resonance by airtone (c) copyright 2020 Licensed under a Creative Commons Attribution Noncommercial (3.0) license. http://dig.ccmixter.org/files/airtone/61321
   - Battle of the Titans by Kraftamt (c) copyright 2020 Licensed under a Creative Commons Attribution Noncommercial (3.0) license. http://dig.ccmixter.org/files/Karstenholymoly/61406 Ft: Snowflake
   - Dawn's Battle (Instrumental) by Ivan Chew (c) copyright 2016 Licensed under a Creative Commons Attribution (3.0) license. http://dig.ccmixter.org/files/ramblinglibrarian/52995
   - Android Battles by Stab (c) copyright 2006 Licensed under a Creative Commons Sampling Plus license. http://dig.ccmixter.org/files/stab/7149
   - the Morning by Vidian (c) copyright 2018 Licensed under a Creative Commons Attribution Noncommercial (3.0) license. http://dig.ccmixter.org/files/Vidian/58453 Ft: Ciggiburns, Aussens@iter, vo1k1

2. *https://www.zapsplat.com/*

The greatest effort has been used to trace the sources of multimedia elements used in the game.

Also, there are several games I have taken inspiration from:

Mincraft, Tower of Saviours, Bookworm Adventures, Pokemon.

Appendix F. Development Timeline (*Game Manual/readme.txt*)

```
/** 1/5/2020 - **/
- Bugfixes:
- NoneType returned when there is no save file. Causes the program to crash due to AttributeError.

/** 1/5/2020 - **/
- The game will be released for students to try out.
- Name list:


/** 25/4/2020 - 30/4/2020 **/
- The game is completed. Testing is now required.
- Finished questions, boss and game design. Formulas have been implemented.
- Ultimate debugging
- Design revamp for improved user friendliness
- Images only load when they are needed -> Performance time is damaged


/** 24/4/2020 **/
"The computer's up an running again!!! after I accidentally cut its power..."
- Finished the new mQuestion system
    - Added nested object attribute call e.g. {1,value,{2,value}}
    - Added function call e.g. {1,valueByNo,10}
- Cannot decide on a database to store the questions

/** 22/4/2020 - 23/4/2020**/
"Some troublshooting, and some more troubleshooting..."
- Completed Multiplayer Mode
    - Through multiprocessing, a player may open a server within a program
      Consider whether making mMultiplayerServer another file is necessary
- Partially completed the new mQuestion
    - Creates object, and coefficient in the statements will refer to the objects
    - Consider implementing checking modes

/** 19/4/2020 - 21/4/2020**/
"The unproductivity is killing me..."
- Started implementing Multiplayer Mode
- Offline Multiplayer is fully available
- Online Multiplayer remains heavily undebugged
- Fading animation added (pyAnimation)
- Revamped AVillage

/** 13/4/2020 **/
"It's the end of the Easter Holiday..."
- Implemented AShop fully
- Implemented AMenu, including all of it's components
- Implemented FrameMenu in pyMenu for creating quick menus
- Implemented ALoad
- Completed Adventure Mode

/** 12/4/2020 **/
- Implemented AVillage, ABossBattle, AGrinding
- Built general layout for AShop: ShopMenu class
- Performance problem solved:
    - Loading images take up too much memory
    - Load images when necessary (Beware of the GlobalVar)
    - Reduce image resolution if required (some are 1900 x 1900)
    - JPEGs does not support transparency

/** 11/4/2020 **/
"Such an unproductive day :("
- Implemented AdventureBattle
    - Serves as the commonplace between Grinding and Boss Battles
    - Includes procedures from choosing escaping to
      choosing potions to dealing damage
```

```
    - Returns 4 types of flags
        - "escape": The player escapes, jump directly out to the village
        - "win": The player defeats the boss
        - "lose": The player is defeated and died
        - "draw": The battle is still not finished -> invoke Main() once more
    - Problems: high memory usage and lagging
- Implemented damage displaying mechanisms in Battle
- Improved display in the Battle class
    - Fits at most 7 lines of text in question (Maybe)
    - Floating arrow to prompt player to press Enter
- Menu now has the scrolling function
(- PERFORMANCE ISSUES
    - The program takes about 70000 K of memory, compared
      with 30000 K in normal execution
    - Constaly blitting images might be the problem
    - Memory test results:
        - Selection Menu : 16000 K
        - Text : 17000 K
         - Text Input: 67320 K / 15904 K
        - Main Menu: 72000 K
        - Battle: 71000 K (in fact, lagging)
        - AdventureBattle: peaks at 79000K)

/** 10/4/2020 **/
- Started implementing battle
    - Health bars
    - Display the player and the opponent
    - Class: Battle
        - Other forms of battles (Multiplayer, Adventure Mode) can
          inhert the Battle class and add their own battle sequences
        - Implemented question answering sequence
- Added GlobalVar (global variables across modules/files)
    - CentralClock
    - Fonts and their filenames (String)
    - Colors (a class accessible by color.___)
- Added pyTimer


/** 9/4/2020 **/
- Implemented AnswerCheck
- Added exception functionality in QuestionGenerator:
    Displays the comment of a question when an error occurs
- Implemented pyInput: TextInput class added
- Implemented PracticeMode
- Implemented Player and load/save modules
    Initialization() --> Player
    LoadSave(filename) --> Player
    SaveFile(filename, player) --> bool


/** Project start to 8/4/2020 **/
- Implemented Main Menu
- Implemented selection menu module
- Implemented text wrapping module
- Implemented QuestionGenerator
    Submodules:
    - QuestionGenerator
    - QuestionReader
```

Appendix G. Game Design

Expectations: answer 20 questions per level, 10 in "Grinding" and 10 in "Journey On…", with "Journey On…" having a correct % of about 75% to 90%[47]. After each level, the player would raise 25 levels, so at end game the player reaches level 75. The bosses' strength and rewards are then calculated based on these expectations.

Experience and Levels

Let N be the experience level. The EXP requirement and skill points awarded for levelling up are calculated as follows:

Item	Formula	N											
		1	**2**	**5**	**10**	**20**	**25**	**30**	**50**	**75**	**100**	**101**	**102** [48]
Amount of EXP required to reach this level X	$X(N)=\left\lceil \frac{100}{3^{\frac{1}{25}}-1}\cdot\left(3^{\frac{N-1}{25}}-1\right)\right\rceil$	0	1000	428	1080	2905	4165	5736	16947	55292	170326	178078	186178
Cumulative EXP up to this level Y	$Y(N)=\sum_{n=1}^{N}X(n)$	0	100	1047	5086	25265	43460	68859	285118	1121366	3741385	3919463	4105641
Skill Points given when this level is reached P	$P(N)=\left\lceil 7.5e^{-\frac{(N-1)}{150}}\right\rceil$		8	8	8	7	7	7	6	5	4	4	4
Cumulative Skill Points p	$p(N)=\sum_{n=2}^{N}P(n)$	0	8	32	72	143	178	213	337	473	593	597	601

[47] This part of the design was not literally documented, so this is an approximation.

[48] Level 102 is hypothetic: it is included here as reference for testing.

Equipment
Suppose x is the equipment tier, where $0 \le x \le 10$. The price and effect of each tier of equipment is given by the following formulae:

Equipment	Item	Formula	x						
			0	**1**	**2**	**3**	**5**	**8**	**10**
Chestplate	Effect: Damage Reduction C	$C(x) = 0.3 \times \frac{x}{10}$	0	3%	6%	9%	15%	24%	30%
	Price c	$c(x) = \left\lceil 8000\left(x\left(1+\frac{x}{10}\right)^{\frac{x}{5}}\right)\right\rceil$		8154	17211	28092	60000	163914	320000
Leggings	Effect: Damage Reduction L	$L(x) = 0.3 \times \frac{x}{10}$	0	3%	6%	9%	15%	24%	30%
	Price l	$l(x) = \left\lceil 6000\left(x\left(1+\frac{x}{10}\right)^{\frac{x}{5}}\right)\right\rceil$		6116	12908	21069	45000	122936	240000
Sword	Effect: Attack Increase S	$S(x) = 0.4 \times \frac{x}{10}$	0	4%	8%	12%	20%	32%	40$
	Price s	$s(x) = \left\lceil 10000\left(x\left(1+\frac{x}{10}\right)^{\frac{x}{5}}\right)\right\rceil$		10193	21514	35115	75000	204893	400000

Potions

Potion of Regenration
$1000
Regenerates 3% of full health.

Potion of Braveness
$4000
Increases attack damage by 5%.

Potions of Perservance
$4000
Reduces opponents' attack damage by 5%.

Skills

Suppose x is the skill level, where $0 \le x \le 200$. The effect of each skill is given by the following formulas:

Skill	Effect	Formula	x											
			0	**1**	**10**	**20**	**30**	**50**	**80**	**100**	**120**	**140**	**170**	**200**
Strength	Base AP A	$A(x) = 650 + \left\lfloor 11350 \times \frac{x}{200} \right\rfloor$	650	706	1217	1785	2352	3487	5190	6325	7460	8595	10297	12000
Health	Base HP H	$H(x) = 900 + \left\lfloor 9100 \times \frac{x}{200} \right\rfloor$	900	945	1355	1810	2265	3175	4540	5450	7725	7270	8635	8545
Defense	Damage Reduction D	$D(x) = 0.6 \times (\frac{x}{200})^{1.2}$	0	0.10%	1.65%	0.037%	6.16%	11.4%	20.0%	26.1%	32.5%	39.1%	49.4%	60%
Agility	Dodge Chance E	$E(x) = 0.3 \times \frac{x}{200}$	0	0.15%	1.50%	3.00%	4.50%	7.50%	12.0%	15.0%	18.0%	21.0%	25.5%	30%
Time Dilation	Time Lengthening T	$T(x) = \frac{2.5\left(1 - e^{-0.76\frac{x}{200}}\right)}{\left(1 - e^{-0.76}\right)}$	0	0.18%	17.5%	34.4%	50.6%	81.3%	123%	148%	172%	194%	223%	250%